

June 13, 2024

SMART CONTRACT AUDIT REPORT

Astrolab DAO
Base Strategy Contracts



omniscia.io



info@omniscia.io



Online report: [astrolab-dao-base-strategy-contracts](#)

Omniscia.io is one of the fastest growing and most trusted blockchain security firms and has rapidly become a true market leader. To date, our team has collectively secured over 370+ clients, detecting 1,500+ high-severity issues in widely adopted smart contracts.

Founded in France at the start of 2020, and with a track record spanning back to 2017, our team has been at the forefront of auditing smart contracts, providing expert analysis and identifying potential vulnerabilities to ensure the highest level of security of popular smart contracts, as well as complex and sophisticated decentralized protocols.

Our clients, ecosystem partners, and backers include leading ecosystem players such as L'Oréal, Polygon, AvaLabs, Gnosis, Morpho, Vesta, Gravita, Olympus DAO, Fetch.ai, and LimitBreak, among others.

To keep up to date with all the latest news and announcements follow us on twitter @omniscia_sec.

 omniscia.io

 info@omniscia.io

Online report: [astrolab-dao-base-strategy-contracts](#)

Base Strategy Contracts Security Audit

Audit Report Revisions

Commit Hash	Date	Audit Report Hash
5427ca2aaa	March 2nd 2024	190edc3e59
59b75fbee1	April 17th 2024	24abe7bc2d
efbeab6478	May 14th 2024	a4b534feab
cf5194da53	June 5th 2024	86d778b017
cf5194da53	June 13th 2024	dfce318558
cf5194da53	June 13th 2024	090dbf4cca

Audit Overview

We were tasked with performing an audit of the Astrolab DAO codebase and in particular their Base Strategy Contracts module.

The project implements a base set of contracts meant to act as the backbone for **EIP-4626** vaults that interact with multiple DeFi protocols via a custom proxy model.

Over the course of the audit, we identified vulnerabilities across multiple modules of the system including incorrect assembly blocks, incorrect downward price action handling, proxy-forwarded data corruption, and more.

The system implements a custom proxy model whereby the Strategy contract and the logic contract are separate, however, this is done so by retaining two different implementations that utilize a shared storage space.

In the current system, the logic contract (`StrategyV5Agent` in this case) will inherit two implementations that declare storage variables while the proxy contract (`StrategyV5`) will inherit three implementations.

This can trivially result in clash of storage space which could ultimately result in data corruption and/or loss.

We recommend the storage of the contracts to be decoupled entirely in a single dedicated implementation, permitting it to be maintained and expanded as required between updates.

We advise the Astrolab DAO team to closely evaluate all minor-and-above findings identified in the report and promptly remediate them as well as consider all optimizational exhibits identified in the report.

Post-Audit Conclusion

The *Post-Audit Conclusion* chapters of the audit report are presented in historical order from oldest to latest. To evaluate the latest state of the codebase, kindly proceed to the last *Post-Audit Conclusion* chapter of the audit report.

The Astrolab DAO team iterated through all findings within the report and provided us with a revised commit hash to evaluate all exhibits on.

We evaluated all alleviations performed by Astrolab DAO and have identified that certain exhibits have not been adequately dealt with. We advise the Astrolab DAO team to revisit the following exhibits which have either been partially alleviated, not alleviated, or incorrectly alleviated: **A62-12M**, **AME-01M**, **ASS-04M**, **CUS-01C**, **ASS-02M**, **A62-08M**, **A62-07M**, **A62-11M**, **SVA-04M**, **AAS-01M**, **AAS-02M**, **SV5-03M**, **PUS-01C**

Additionally, the following *informational* findings remain either partially addressed or unaddressed and should be revisited: **ASS-02C**, **ASS-01C**, **ASS-03C**, **AME-02C**, **AMS-01C**, **AMS-04C**, **AMS-02C**, **AMS-03C**, **ARA-01C**, **ARE-04C**, **ARE-02C**, **SVA-01C**, **AAS-02C**, **SV5-02C**, **SV5-04C**, **SV5-06C**

Post-Audit Conclusion (efbeab6478)

The Astrolab DAO team provided us with a follow-up commit to evaluate additional remediations carried out for the instances that remained open in the previous round, as well as general adjustments in relation to the **EIP-7540** compliancy of the `As4626` implementation.

We observed that exhibit **A62-11M** which concerns **EIP-7540** compliancy is still not resolved despite the change in the project's direction to solely support redemption requests as the EIP is still not satisfied in this regard.

In addition to the aforementioned exhibits that remain open, the following exhibits have been marked as acknowledged explicitly by the Astrolab DAO team: **AME-01M**, **AME-02C**, **ASS-01C**, **ASS-02C**, **ASS-03C**, **AAS-02C**, **SV5-06C**, **AMS-01C**, **AMS-02C**, **AMS-03C**, **AMS-04C**, **ARA-01C**, **ARE-02C**, **ARE-04C**, **SV5-03M**

Finally, in between the production of the previous final iteration and the current version, we came in contact with the Pyth Network team to clarify what limitations should be imposed on their oracles.

The Pyth Network team contradicted the SDK implementation and instead clarified that the exponents supported by the Pyth Network oracle software are within the following range: `[-12, 12]`

In light of this information, we advise the `PythProvider::toUsdBp` function to be updated with those exponents in mind properly supporting positive as well as negative exponents which it presently does not.

Post-Audit Conclusion (cf5194da53)

The Astrolab DAO team revisited a subset of the exhibits mentioned in the previous chapter; namely:

A62-11M, **AAS-02C**, **AMS-02C**, **AMS-04C**, **ARE-02C**

All aforementioned exhibits have been properly alleviated in the latest commit hash of the codebase that was evaluated, and any that were not mentioned have been marked as acknowledged.

Additionally, the `PythProvider` related concerns have been addressed by incorporating support for positive exponents as well as adjusting the range of permitted exponent values.

We consider all outputs of the audit report properly consumed by the Astrolab DAO team, and no further remediative actions are expected.

Audit Synopsis

Severity	Identified	Alleviated	Partially Alleviated	Acknowledged
Unknown	0	0	0	0
Informational	64	52	0	12
Minor	23	22	0	1
Medium	0	0	0	0
Major	9	9	0	0

During the audit, we filtered and validated a total of **7 findings utilizing static analysis** tools as well as identified a total of **89 findings during the manual review** of the codebase. We strongly recommend that any minor severity or higher findings are dealt with promptly prior to the project's launch as they can introduce potential misbehaviours of the system as well as exploits.

Scope

The audit engagement encompassed a specific list of contracts that were present in the commit hash of the repository that was in scope. The tables below detail certain meta-data about the target of the security assessment and a navigation chart is present at the end that links to the relevant findings per file.

Target

- Repository: <https://github.com/AstrolabDAO/strats>
- Commit: 5427ca2aaaafa0be3b90fc057a8b79f4088cba32
- Language: Solidity
- Network: arbitrum, optimism, base, polygon, linea, scroll, mantle, gnosis, moonbeam
- Revisions: 5427ca2aaa, 59b75fbee1, efbeab6478, cf5194da53

Contracts Assessed

File	Total Finding(s)
src/abstract/As4626.sol (A62)	17
src/libs/AsCast.sol (ACT)	1
src/libs/AsMaths.sol (AMS)	6
src/abstract/AsProxy.sol (APY)	4
src/abstract/AsTypes.sol (ATS)	0
src/libs/AsArrays.sol (AAS)	7
src/abstract/AsRescuable.sol (ARE)	6
src/libs/AsAccounting.sol (AAG)	1
src/abstract/AsManageable.sol (AME)	5
src/abstract/As4626Abstract.sol (AAT)	4

src/abstract/AsAccessControl.sol (AAC)	3
src/libs/AsSequentialSet.sol (ASS)	9
src/abstract/AsRescuableAbstract.sol (ARA)	1
src/libs/ChainlinkUtils.sol (CUS)	2
src/libs/PythUtils.sol (PUS)	2
src/abstract/StrategyV5.sol (SV5)	11
src/abstract/StrategyV5Pyth.sol (SVP)	3
src/abstract/StrategyV5Agent.sol (SVA)	7
src/abstract/StrategyV5Abstract.sol (SVT)	1
src/abstract/StrategyV5Chainlink.sol (SVC)	6

Compilation

The project utilizes `hardhat` as its development pipeline tool, containing an array of tests and scripts coded in TypeScript.

To compile the project, the `compile` command needs to be issued via the `npx` CLI tool to `hardhat`:

BASH

```
npx hardhat compile
```

The `hardhat` tool automatically selects Solidity version `0.8.22` based on the version specified within the `hardhat.config.ts` file.

The project contains discrepancies with regards to the Solidity version used as the `pragma` statements of the contracts are open-ended (`^0.8.0`).

We advise them to be locked to `0.8.22` (`=0.8.22`), the same version utilized for our static analysis as well as optimizational review of the codebase.

During compilation with the `hardhat` pipeline, no errors were identified that relate to the syntax or bytecode size of the contracts.

To note, the compiler version utilized makes use of the Shanghai target EVM and thus will introduce the `PUSH0` opcode which is incompatible with certain Layer-2 chains.

We advise the Astrolab DAO team to evaluate whether the chains they wish to deploy their contracts to properly support the operation code when they intend to deploy so as to avoid any deployment failures and thus waste of resources.

Static Analysis

The execution of our static analysis toolkit identified **83 potential issues** within the codebase of which **67 were ruled out to be false positives** or negligible findings.

The remaining **16 issues** were validated and grouped and formalized into the **7 exhibits** that follow:

ID	Severity	Addressed	Title
A62-01S	Informational	✓ Yes	Inexistent Event Emissions
AAT-01S	Informational	✓ Yes	Inexistent Event Emission
AMS-01S	Informational	✓ Yes	Illegible Numeric Value Representation
ARE-01S	Informational	✓ Yes	Inexistent Visibility Specifiers
ARE-02S	Minor	✓ Yes	Deprecated Native Asset Transfer
SV5-01S	Informational	✓ Yes	Inexistent Event Emission
SVC-01S	Informational	✓ Yes	Inexistent Event Emission

Manual Review

A **thorough line-by-line review** was conducted on the codebase to identify potential malfunctions and vulnerabilities in Astrolab DAO's base strategy contracts.

As the project at hand implements custom proxies w/ extensive assembly blocks, intricate care was put into ensuring that the **flow of funds within the system conforms to the specifications and restrictions** laid forth within the protocol's specification and that the EVM's restrictions are adhered to in all statements.

We validated that **all state transitions of the system occur within sane criteria** and that all rudimentary formulas within the system execute as expected. We **pinpointed multiple significant vulnerabilities** within the system which could have had **severe ramifications** to its overall operation; we urge the Astrolab DAO team to promptly evaluate and remediate them.

Additionally, the system was investigated for any other commonly present attack vectors such as re-entrancy attacks, mathematical truncations, logical flaws and **ERC / EIP** standard inconsistencies. The documentation of the project was satisfactory to the extent it need be, however, certain areas of the codebase such as expected EIP-7540 conformity should be expanded upon.

A total of **89 findings** were identified over the course of the manual review of which **36 findings** concerned the behaviour and security of the system. The non-security related findings, such as optimizations, are included in the separate **Code Style** chapter.

The finding table below enumerates all these security / behavioural findings:

ID	Severity	Addressed	Title
A62-01M	Minor	Yes	Discrepancy of Access Control
A62-02M	Minor	Yes	Improper Allowance Adjustment
A62-03M	Minor	Yes	Improper Capture of Entry Fee
A62-04M	Minor	Yes	Improper Capture of Exit Fee
A62-05M	Minor	Yes	Incorrect Estimation of Deposits

A62-06M	Minor	Yes	Incorrect Estimation of Withdrawals
A62-07M	Minor	Yes	Incorrect Maintenance of Allowances in Redemption Requests
A62-08M	Minor	Yes	Inexistent Protection Against Re-Initialization
A62-09M	Minor	Yes	Potentially Invalid Cancellation Assumption
A62-10M	Major	Yes	Improper Accounting of Fees in Downward Price Action
A62-11M	Major	Yes	Incorrect Implementation of EIP-7540
A62-12M	Major	Yes	Inexistent Reservation of Shares
AAT-01M	Minor	Yes	EIP-7540 Incompatibility
AAS-01M	Major	Yes	Incorrect EVM Memory Assumptions

AAS-02M	● Major	✓ Yes	Incorrect Usage of Memory
ACT-01M	● Minor	✓ Yes	Potentially Insecure Address Cast
AME-01M	● Informational	! Acknowledged	Invalid Conditional Evaluation
AME-02M	● Major	✓ Yes	Detachment of Authorized Role
AMS-01M	● Informational	✓ Yes	Improper Absolute Function Implementation
APY-01M	● Informational	🚫 Nullified	Reservation of Function Signatures
APY-02M	● Minor	🚫 Nullified	Potentially Insecure Utilization of Scratch Space
APY-03M	● Major	🚫 Nullified	Insecure Forwarded Payload
ASS-01M	● Informational	✓ Yes	Improper Sequential Set Shift Operation

ASS-02M	Minor	Yes	Inexistent Prevention of Duplicate Elements
ASS-03M	Major	Yes	Invalid Sequential Set Shift Operation
ASS-04M	Major	Yes	Invalid Sequential Set Unshift Operation
SV5-01M	Informational	Yes	Implementation & Documentation Mismatch
SV5-02M	Minor	Yes	Discrepancy of Liquidation Preview
SV5-03M	Minor	Acknowledged	Insecure Casting Operations
SVA-01M	Minor	Yes	Discrepant Allowance Maintenance
SVA-02M	Minor	Yes	Improper No-Op Logic Statement
SVA-03M	Minor	Yes	Inexistent Erasure of Previous Approvals

SVA-04M	Minor	Yes	Inexistent Protection Against Re-Initialization
SVA-05M	Minor	Yes	Insecure Approval Operations
SVC-01M	Minor	Yes	Inexistent Prevention of Data Corruption
SVC-02M	Minor	Yes	Inexistent Validation of Prices

Code Style

During the manual portion of the audit, we identified **53 optimizations** that can be applied to the codebase that will decrease the operational cost associated with the execution of a particular function and generally ensure that the project complies with the latest best practices and standards in Solidity.

Additionally, this section of the audit contains any opinionated adjustments we believe the code should make to make it more legible as well as truer to its purpose.

These optimizations are enumerated below:

ID	Severity	Addressed	Title
A62-01C	● Informational	✓ Yes	Inefficient <code>mapping</code> Lookups
A62-02C	● Informational	✗ Nullified	Redundant Duplication of Code
A62-03C	● Informational	✓ Yes	Redundant Parenthesis Statements
A62-04C	● Informational	✓ Yes	Repetitive Value Literal
AAT-01C	● Informational	✓ Yes	Generic Typographic Mistakes
AAT-02C	● Informational	✓ Yes	Improper Declaration of Abstract Function
AAC-01C	● Informational	✓ Yes	Inefficient Usage of Utility Functions
AAC-02C	● Informational	✓ Yes	Redundant Input Argument
AAC-03C	● Informational	✓ Yes	Redundant Local Variable
AAG-01C	● Informational	✓ Yes	Repetitive Value Literal

AAS-01C	● Informational	✖ Nullified	Ineffectual Usage of Safe Arithmetics
AAS-02C	● Informational	✓ Yes	Inefficient Iteration of Search Loops
AAS-03C	● Informational	✓ Yes	Inefficient Iterator Type
AAS-04C	● Informational	✓ Yes	Inexistent Error Messages
AAS-05C	● Informational	✓ Yes	Loop Iterator Optimizations
AME-01C	● Informational	✓ Yes	Generic Typographic Mistakes
AME-02C	● Informational	❗ Acknowledged	Inexistent Error Message
AME-03C	● Informational	✓ Yes	Redundant Parenthesis Statements
AMS-01C	● Informational	❗ Acknowledged	Generic Typographic Mistakes
AMS-02C	● Informational	✓ Yes	Ineffectual Usage of Safe Arithmetics
AMS-03C	● Informational	❗ Acknowledged	Inexistent Error Messages
AMS-04C	● Informational	✓ Yes	Redundant Parenthesis Statements
APY-01C	● Informational	✖ Nullified	Inefficient Generation of Selector
ARE-01C	● Informational	✖ Nullified	Improper Declarations of Abstract Functions

ARE-02C	Informational	Yes	Inefficient Erasure of Request
ARE-03C	Informational	Yes	Inefficient mapping Lookups
ARE-04C	Informational	Acknowledged	Inexistent Error Messages
ARA-01C	Informational	Acknowledged	Optimization of Data Structure
ASS-01C	Informational	Acknowledged	Ineffectual Usage of Safe Arithmetics
ASS-02C	Informational	Acknowledged	Inefficient Loop Limit Evaluations
ASS-03C	Informational	Acknowledged	Inexistent Error Message
ASS-04C	Informational	Yes	Loop Iterator Optimization
ASS-05C	Informational	Yes	Redundant Deletion Operation
CUS-01C	Informational	Acknowledged	Ineffectual Usage of Safe Arithmetics
CUS-02C	Informational	Yes	Repetitive Value Literal
PUS-01C	Informational	Acknowledged	Ineffectual Usage of Safe Arithmetics
PUS-02C	Informational	Yes	Repetitive Value Literal
SV5-01C	Informational	Yes	Generic Typographic Mistake

SV5-02C	Informational	✓ Yes	Improper Declarations of Abstract Functions
SV5-03C	Informational	✓ Yes	Ineffectual Usage of Safe Arithmetics
SV5-04C	Informational	✓ Yes	Inefficient Iterator Type
SV5-05C	Informational	✓ Yes	Loop Iterator Optimizations
SV5-06C	Informational	⚠ Acknowledged	Redundant Application of Access Control
SV5-07C	Informational	✓ Yes	Redundant Parenthesis Statement
SVT-01C	Informational	🚫 Nullified	Generic Typographic Mistakes
SVA-01C	Informational	✓ Yes	Inefficient Iterator Type
SVA-02C	Informational	✓ Yes	Loop Iterator Optimizations
SVC-01C	Informational	✓ Yes	Generic Typographic Mistake
SVC-02C	Informational	✓ Yes	Loop Iterator Optimization
SVC-03C	Informational	✓ Yes	Repetitive Value Literal
SVP-01C	Informational	✓ Yes	Generic Typographic Mistake
SVP-02C	Informational	✓ Yes	Loop Iterator Optimizations

SVP-03C

 Informational

 Yes

Repetitive Value Literal

As4626 Static Analysis Findings

A62-01S: Inexistent Event Emissions

Type	Severity	Location
Language Specific	Informational	As4626.sol:L353-L356, L362-L364, L370-L372, L378-L380, L4

Description:

The linked functions adjust sensitive contract variables yet do not emit an event for it.

Example:

```
src/abstract/As4626.sol
```

```
SOL
```

```
362 function setMaxSlippageBps(uint16 _slippageBps) external onlyManager {  
363     maxSlippageBps = _slippageBps;  
364 }
```

Recommendation:

We advise an `event` to be declared and correspondingly emitted for each function to ensure off-chain processes can properly react to this system adjustment.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The Astrolab DAO team evaluated this exhibit and specified that they have consciously removed certain event emissions due to their impact on the bytecode size of the contracts.

In light of this issue, critical events have been selectively re-introduced where possible in compliance with the bytecode size limitations of the blockchain the contracts are deployed in.

As such, we consider this exhibit addressed to the greatest extent possible when acknowledging EVM related constraints.

As4626Abstract Static Analysis Findings

AAT-01S: Inexistent Event Emission

Type	Severity	Location
Language Specific	● Informational	As4626Abstract.sol:L98-L100

Description:

The linked function adjusts a sensitive contract variable yet does not emit an event for it.

Example:

```
src/abstract/As4626Abstract.sol
```

```
SOL
```

```
98 function setExemption(address _account, bool _isExempt) public onlyAdmin {  
99     exemptionList[_account] = _isExempt;  
100 }
```

Recommendation:

We advise an `event` to be declared and correspondingly emitted to ensure off-chain processes can properly react to this system adjustment.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The Astrolab DAO team evaluated this exhibit and specified that they have consciously removed certain event emissions due to their impact on the bytecode size of the contracts.

In light of this issue, critical events have been selectively re-introduced where possible in compliance with the bytecode size limitations of the blockchain the contracts are deployed in.

As such, we consider this exhibit addressed to the greatest extent possible when acknowledging EVM related constraints.

AsMaths Static Analysis Findings

AMS-01S: Illegible Numeric Value Representation

Type	Severity	Location
Code Style	● Informational	AsMaths.sol:L22

Description:

The linked representation of a numeric literal is sub-optimally represented decreasing the legibility of the codebase.

Example:

```
src/libs/AsMaths.sol
```

```
SOL
```

```
22 uint256 internal constant BP_BASIS = 10_000; // 50% == 5_000 == 5e3
```

Recommendation:

To properly illustrate the value's purpose, we advise the following guidelines to be followed. For values meant to depict fractions with a base of `1e18`, we advise fractions to be utilized directly (i.e. `1e17` becomes `0.1e18`) as they are supported. For values meant to represent a percentage base, we advise each value to utilize the underscore (`_`) separator to discern the percentage decimal (i.e. `10000` becomes `100_00`, `300` becomes `3_00` and so on). Finally, for large numeric values we simply advise the underscore character to be utilized again to represent them (i.e. `1000000` becomes `1_000_000`).

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The referenced value literal has been updated in its representation to `100_00` in accordance with the recommendation's underscore style, addressing this exhibit.

AsRescuable Static Analysis Findings

ARE-01S: Inexistent Visibility Specifiers

Type	Severity	Location
Code Style	Informational	AsRescuable.sol:L21, L22

Description:

The linked variables have no visibility specifier explicitly set.

Example:

```
src/abstract/AsRescuable.sol
SOL
21 uint64 constant RESCUE_TIMELOCK = 2 days;
```

Recommendation:

We advise them to be set so to avoid potential compilation discrepancies in the future as the current behaviour is for the compiler to assign one automatically which may deviate between `pragma` versions.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The `public` visibility specifier has been introduced to all referenced variables, preventing potential compilation discrepancies and addressing this exhibit.

ARE-02S: Deprecated Native Asset Transfer

Type	Severity	Location
Language Specific	Minor	AsRescuable.sol:L88

Description:

The linked statement performs a low-level native asset transfer via the `transfer` function exposed by the `address payable` data type.

Impact:

As new EIPs such as **EIP-2930** are introduced to the blockchain, gas costs can change and the `transfer` instruction of Solidity specifies a fixed gas stipend that is prone to failure should such changes be integrated to the blockchain the contract is deployed in. A prime example of this behaviour are legacy versions of Gnosis which were susceptible to this issue and would cause native transfers to fail if sent to a new address.

Example:

```
src/abstract/AsRescuable.sol
SOL
88 payable(req.receiver).transfer(address(this).balance);
```

Recommendation:

We advise alternative ways of transferring assets to be utilized instead, such as OpenZeppelin's `Address.sol` library and in particular the `sendValue` method exposed by it. If re-entrancies are desired to be prevented based on gas costs, we instead advise a mechanism to be put in place that either credits an account with a native balance they can withdraw at a secondary transaction or that performs the native asset transfers at the end of the top-level transaction's execution.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The native payment has been replaced by a low-level `call` interaction that supplies the full available gas allowance to the call thus ensuring it will succeed regardless of the underlying blockchain the contract is deployed in or the nature of the recipient.

StrategyV5 Static Analysis Findings

SV5-01S: Inexistent Event Emission

Type	Severity	Location
Language Specific	Informational	StrategyV5.sol:L98-L101

Description:

The linked function adjusts a sensitive contract variable yet does not emit an event for it.

Example:

```
src/abstract/StrategyV5.sol
```

```
SOL
```

```
98  function updateAgent(address _agent) external onlyAdmin {
99      if (_agent == address(0)) revert AddressZero();
100     agent = _agent;
101 }
```

Recommendation:

We advise an `event` to be declared and correspondingly emitted to ensure off-chain processes can properly react to this system adjustment.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The Astrolab DAO team evaluated this exhibit and specified that they have consciously removed certain event emissions due to their impact on the bytecode size of the contracts.

In light of this issue, critical events have been selectively re-introduced where possible in compliance with the bytecode size limitations of the blockchain the contracts are deployed in.

As such, we consider this exhibit addressed to the greatest extent possible when acknowledging EVM related constraints.

StrategyV5Chainlink Static Analysis Findings

SVC-01S: Inexistent Event Emission

Type	Severity	Location
Language Specific	Informational	StrategyV5Chainlink.sol:L55-L59

Description:

The linked function adjusts a sensitive contract variable yet does not emit an event for it.

Example:

```
src/abstract/StrategyV5Chainlink.sol
SOL
55  function setPriceFeed(address _address, IChainlinkAggregatorV3 _feed, uint256
56    _validity) public onlyAdmin {
57      feedByAsset[_address] = _feed;
58      decimalsByFeed[_feed] = feedByAsset[_address].decimals();
59      validityByFeed[feedByAsset[_address]] = _validity;
59 }
```

Recommendation:

We advise an `event` to be declared and correspondingly emitted to ensure off-chain processes can properly react to this system adjustment.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The Astrolab DAO team evaluated this exhibit and specified that they have consciously removed certain event emissions due to their impact on the bytecode size of the contracts.

In light of this issue, critical events have been selectively re-introduced where possible in compliance with the bytecode size limitations of the blockchain the contracts are deployed in.

As such, we consider this exhibit addressed to the greatest extent possible when acknowledging EVM related constraints.

As4626 Manual Review Findings

A62-01M: Discrepancy of Access Control

Type	Severity	Location
Logical Fault	Minor	As4626.sol:L231, L249, L266, L284-L289

Description:

The `As4626::withdraw` and `As4626::redeem` functions prevent invocation if the `_owner` of the shares being withdrawn is not the `msg.sender`, however, their `safe`-prefixed counterparts do not perform such validation.

Additionally, the `As4626::_withdraw` implementation properly supports allowance consumptions so the presence of access control is contradictory.

Impact:

While the discrepancy itself does not result in any vulnerability due to proper allowance management in the `As4626::_withdraw` function, we still consider it to be a non-informational issue in the code as it could have had a significant impact to its security.

Example:

```
src/abstract/As4626.sol
```

```
SOL  
218 /**
219  * @notice Withdraw by burning the equivalent _owner's shares and sending _amount
220  * @dev Beware, there's no slippage control - use safeWithdraw if you want it
221  * @param _amount Amount of asset tokens to withdraw
222  * @param _receiver Who will get the withdrawn assets
223  * @param _owner Whose shares we'll burn
224  * @return shares Amount of shares burned
225 */
226 function withdraw(
227     uint256 _amount,
```

Example (Cont.):

SOL

```
228     address _receiver,
229     address _owner
230 ) external whenNotPaused returns (uint256) {
231     if (_owner != msg.sender) revert Unauthorized();
232     return _withdraw(_amount, previewWithdraw(_amount, _owner), _receiver,
233 _owner);
234
235 /**
236  * @notice Withdraw assets denominated in asset
237  * @dev Overloaded version with slippage control
238  * @param _amount Amount of asset tokens to withdraw
239  * @param _receiver Who will get the withdrawn assets
240  * @param _owner Whose shares we'll burn
241  * @return amount Amount of shares burned
242 */
243 function safeWithdraw(
244     uint256 _amount,
245     uint256 _minAmount,
246     address _receiver,
247     address _owner
248 ) public whenNotPaused returns (uint256 amount) {
249     amount = _withdraw(_amount, previewWithdraw(_amount, _owner), _receiver,
250 _owner);
250     if (amount < _minAmount) revert AmountTooLow(amount);
251 }
```

Recommendation:

We advise access control to either be imposed on all variants of these functions or to be omitted entirely, either of which we consider an adequate resolution to this exhibit.

Alleviation (59b75fbee1d8f3dee807c928f18be41c58b904e1):

Both functions and their `safe`-prefixed counterparts now properly support allowance-based authorization of the `msg.sender`, ensuring uniform behaviour across the functions by addressing this exhibit.

A62-02M: Improper Allowance Adjustment

Type	Severity	Location
Logical Fault	Minor	As4626.sol:L690-L694

Description:

An `As4626::cancelRedeemRequest` operation will consume the allowance between the `_operator` and the `_owner` only if the `opportunityCost` is greater than `0`, however, the adjustment will be for the full `shares` amount.

Impact:

The present mechanism will most likely consume a higher allowance than it should incorrectly.

Example:

src/abstract/As4626.sol

```
SOL

690 // Adjust the operator's allowance after burning shares, only if the operator is
different from the owner
691 if (opportunityCost > 0 && _owner != msg.sender) {
692     uint256 currentAllowance = allowance(_owner, _operator);
693     _approve(_owner, _operator, currentAllowance - shares);
694 }
```

Recommendation:

We advise this approach to be revised as it is presently invalid. The code should either revoke an approval equivalent to the `opportunityCost`, or should unconditionally revoke the full approval of the `shares`.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The approval adjustment properly utilizes the `opportunityCost` in the latest implementation, addressing this exhibit.

A62-03M: Improper Capture of Entry Fee

Type	Severity	Location
Mathematical Operations	Minor	As4626.sol:L130

Description:

Any deposit-related function that utilizes `As4626::previewDeposit` will suffer truncation in contrast to the `As4626::previewMint` function as the basis point percentages are applied in a rounding-prone way.

Impact:

Fees captured from deposits and their respective deposit amount may not sum up to the actual amount the user supplied due to truncation.

Example:

```
src/abstract/As4626.sol
SOL
466 /**
467 * @notice Previews the amount of shares that will be minted for a given deposit
amount
468 * @param _amount Amount of asset tokens to deposit
469 * @param _receiver The future owner of the shares to be minted
470 * @return shares Amount of shares that will be minted
471 */
472 function previewDeposit(uint256 _amount, address _receiver) public view returns
(uint256 shares) {
473     return convertToShares(_amount, false).subBp(exemptionList[_receiver] ? 0 :
fees.entry);
474 }
```

Recommendation:

The flaw arises from the following misconception:

$$x * 100$$

We advise the basis-point related calculations to be streamlined across the codebase, ensuring that truncation is accounted for by utilizing the remainder of the amount after the fee's application.

Alleviation (59b75fbee1d8f3dee807c928f18be41c58b904e1):

The code has been refactored to no longer use the `preview`-related functions in the input `_shares` of the `As4626::deposit` function, calculating the fee locally instead.

In this implementation, the actual deposited amount is calculated as the original amount minus the fee captured, ensuring that any truncation which may occur is solely reflected in the fee and does not impact the deposited amount.

As such, we consider this exhibit fully alleviated.

A62-04M: Improper Capture of Exit Fee

Type	Severity	Location
Mathematical Operations	Minor	As4626.sol:L267, L285

Description:

Any withdrawal-related function that utilizes `As4626::previewRedeem` will suffer truncation in contrast to the `As4626::previewWithdraw` function as the basis point percentages are applied in a rounding-prone way.

Impact:

Fees captured from withdrawals and their respective withdrawal amount may not sum up to the actual amount that the user is entitled to due to truncation.

Example:

```
src/abstract/As4626.sol
```

```
SOL

253 /**
254  * @notice Redeems/burns _owner's shares and sends the equivalent amount in asset
255  * to _receiver
256  * @dev Beware, there's no slippage control - you need to use the overloaded
257  * function if you want it
258  * @param _shares Amount of shares to redeem
259  * @param _receiver Who will get the withdrawn assets
260  * @param _owner Whose shares we'll burn
261  * @return assets Amount of assets withdrawn
262 */
263 function redeem(
264     uint256 _shares,
```

Example (Cont.):

```
SOL

263     address _receiver,
264     address _owner
265 ) external whenNotPaused returns (uint256 assets) {
266     if (_owner != msg.sender) revert Unauthorized();
267     return _withdraw(previewRedeem(_shares, _owner), _shares, _receiver, _owner);
268 }
269
270 /**
271  * @dev Overloaded version with slippage control
272  * @param _shares Amount of shares to redeem
273  * @param _minAmountOut The minimum amount of assets accepted
274  * @param _receiver Who will get the withdrawn assets
275  * @param _owner Whose shares we'll burn
276  * @return assets Amount of assets withdrawn
277 */
278 function safeRedeem(
279     uint256 _shares,
280     uint256 _minAmountOut,
281     address _receiver,
282     address _owner
283 ) external whenNotPaused returns (uint256 assets) {
284     assets = _withdraw(
285         previewRedeem(_shares, _owner),
286         _shares, // _shares
287         _receiver, // _receiver
288         _owner // _owner
289     );
290     if (assets < _minAmountOut) revert AmountTooLow(assets);
```

Example (Cont.):

SOL

291 }

Recommendation:

The flaw arises from the following misconception:

$$x * 100$$

We advise the basis-point related calculations to be streamlined across the codebase, ensuring that truncation is accounted for by utilizing the remainder of the amount after the fee's application.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The code has been refactored to no longer use the `preview`-related functions in the input `_amount` of the `As4626::withdraw` function, calculating the fee locally instead.

In this implementation, the actual withdrawn amount is calculated as the original amount minus the fee captured, ensuring that any truncation which may occur is solely reflected in the fee and does not impact the withdrawn amount.

As such, we consider this exhibit fully alleviated.

A62-05M: Incorrect Estimation of Deposits

Type	Severity	Location
Mathematical Operations	Minor	As4626.sol:L452, L473

Description:

The `As4626::previewDeposit` function will incorrectly estimate the amount of shares the operation will result in as it will apply the entry fee **on the shares minted** rather than **the tokens deposited**, causing truncation issues to lead to different results.

Impact:

As shares will most likely have a lower accuracy than the assets deposited, the truncation will be more severe and thus underestimate the amount of shares that will be minted.

Example:

```
src/abstract/As4626.sol

SOL

466 /**
467 * @notice Previews the amount of shares that will be minted for a given deposit
amount
468 * @param _amount Amount of asset tokens to deposit
469 * @param _receiver The future owner of the shares to be minted
470 * @return shares Amount of shares that will be minted
471 */
472 function previewDeposit(uint256 _amount, address _receiver) public view returns
(uint256 shares) {
473     return convertToShares(_amount, false).subBp(exemptionList[_receiver] ? 0 :
fees.entry);
474 }
```

Recommendation:

We advise the code to properly apply the entry fee to the input `_amount`, simulating the behaviour of the `As4626::deposit` function.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The entry fee is correctly applied to the input `_amount` of the `As4626::previewDeposit` function in the latest implementation, addressing this exhibit.

A62-06M: Incorrect Estimation of Withdrawals

Type	Severity	Location
Mathematical Operations	Minor	As4626.sol:L495, L516

Description:

The `As4626::previewWithdraw` function will incorrectly estimate the amount of shares the operation will burn as it will apply the exit fee **on the shares burned** rather than **the tokens withdrawn**, causing truncation issues to lead to different results.

Impact:

As shares will most likely have a lower accuracy than the assets deposited, the truncation will be more severe and thus overestimate the amount of shares that will be burned.

Example:

```
src/abstract/As4626.sol
```

```
SOL
```

```
487 /**
488  * @notice Preview how many shares the caller needs to burn to get his assets
back
489  * @dev You may get less asset tokens than you expect due to slippage
490  * @param _assets How much we want to get
491  * @param _owner The owner of the shares to be redeemed
492  * @return How many shares will be burnt
493 */
494 function previewWithdraw(uint256 _assets, address _owner) public view returns
(uint256) {
495     return convertToShares(_assets, true).revAddBp(exemptionList[_owner] ? 0 :
fees.exit);
496 }
```

Recommendation:

We advise the code to properly apply the exit fee to the output `_amount`, simulating the behaviour of the `As4626::withdraw` function.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The exit fee is correctly applied to the input `_amount` of the `As4626::previewWithdraw` function in the latest implementation, addressing this exhibit.

A62-07M: Incorrect Maintenance of Allowances in Redemption Requests

Type	Severity	Location
Logical Fault	Minor	As4626.sol:L179, L670, L693

Description:

The creation of a redemption request will correctly ensure the caller (i.e. `_operator`) has been authorized to create the request (in case they are not the `_owner` themselves), however, the same approval will be incorrectly validated for cancellations as well as processing of these requests.

Impact:

A redemption request that was planned for another user can be trivially hijacked by the original `_owner` if they revoke their allowance, a trait we consider invalid in the system albeit with a small consequence.

Example:

```
src/abstract/As4626.sol
SOL
579 /**
580  * @notice Initiate a redeem request for shares
581  * @param _shares Amount of shares to redeem
582  * @param _operator Address initiating the request
583  * @param _owner The owner of the shares to be redeemed
584 */
585 function requestRedeem(
586     uint256 _shares,
587     address _operator,
588     address _owner,
```

Example (Cont.):

```
SOL

589     bytes memory _data
590 ) public nonReentrant whenNotPaused returns (uint256 _requestId) {
591     if (_operator != msg.sender || (_owner != msg.sender && allowance(_owner,
592     _operator) < _shares))
593         revert Unauthorized();
594     if (_shares == 0 || balanceOf(_owner) < _shares)
595         revert AmountTooLow(_shares);
596
597     Erc7540Request storage request = req.byOwner[_owner];
598     if (request.operator != _operator) request.operator = _operator;
599
600     last.sharePrice = sharePrice();
601     if (request.shares > 0) {
602         if (request.shares > _shares)
603             revert AmountTooLow(_shares);
604
605         // reinit the request (re-added lower)
606         req.totalRedemption -= AsMaths.min(
607             req.totalRedemption,
608             request.shares
609         );
610         // compute request vwap
611         request.sharePrice =
612             ((last.sharePrice * (_shares - request.shares)) + (request.sharePrice
613             * request.shares)) /
614             _shares;
615     } else {
616         request.sharePrice = last.sharePrice;
617     }
618 }
```

Example (Cont.):

```
SOL

617     _requestId = ++requestId;
618     request.requestId = _requestId;
619     request.shares = _shares;
620     request.timestamp = block.timestamp;
621     req.totalRedemption += _shares;
622
623     if(_data.length != 0) {
624         // the caller contract must return the bytes4 value "0x0102fde4"
625         if(IERC7540RedeemReceiver(msg.sender).onERC7540RedeemReceived(_operator,
626             _owner, _requestId, _data) != 0x0102fde4)
627             revert Unauthorized();
628     }
629     emit RedeemRequest(_owner, _operator, _owner, _shares);
630
631 /**
632 * @notice Initiate a withdraw request for assets denominated in asset
633 * @param _amount Amount of asset tokens to withdraw
634 * @param _operator Address initiating the request
635 * @param _owner The owner of the shares to be redeemed
636 * @param _data Additional data
637 * @return requestId The ID of the withdraw request
638 */
639 function requestWithdraw(
640     uint256 _amount,
641     address _operator,
642     address _owner,
643     bytes memory _data
644 ) external returns (uint256) {
```

Example (Cont.):

SOL

```
645     return requestRedeem(convertToShares(_amount, false), _operator, _owner,
646     _data);
647
648 // /**
649 //   * @notice Cancel a deposit request
650 //   * @param operator Address initiating the request
651 //   * @param owner The owner of the shares to be redeemed
652 //   */
653 // function cancelDepositRequest(
654 //     address operator,
655 //     address owner
656 // ) external virtual nonReentrant {}
657
658 /**
659  * @notice Cancel a redeem request
660  * @param _operator Address initiating the request
661  * @param _owner The owner of the shares to be redeemed
662  */
663 function cancelRedeemRequest(
664     address _operator,
665     address _owner
666 ) external nonReentrant {
667     Erc7540Request storage request = req.byOwner[_owner];
668     uint256 shares = request.shares;
669
670     if (_operator != msg.sender || (_owner != msg.sender && allowance(_owner,
671         _operator) < shares))
671         revert Unauthorized();
672     if (shares == 0) revert AmountTooLow(0);
```

Example (Cont.):

```
SOL

673
674     last.sharePrice = sharePrice();
675     uint256 opportunityCost = 0;
676     if (last.sharePrice > request.sharePrice) {
677         // burn the excess shares from the loss incurred while not farming
678         // with the idle funds (opportunity cost)
679         opportunityCost = shares.mulDiv(
680             last.sharePrice - request.sharePrice,
681             weiPerShare
682         ); // eg. 1e8+1e8-1e8 = 1e8
683         _burn(_owner, opportunityCost);
684     }
685
686     req.totalRedemption -= shares;
687     if (isRequestClaimable(request.timestamp))
688         req.totalClaimableRedemption -= shares;
689
690     // Adjust the operator's allowance after burning shares, only if the operator
691     // is different from the owner
692     if (opportunityCost > 0 && _owner != msg.sender) {
693         uint256 currentAllowance = allowance(_owner, _operator);
694         _approve(_owner, _operator, currentAllowance - shares);
695     }
696     request.shares = 0;
697     emit RedeemRequestCanceled(_owner, shares);
698 }
```

Recommendation:

We advise the code to consume the allowance during a redemption request's creation, and to permit the creator of the request (i.e. caller of `As4626::requestRedeem`) to either cancel or claim the request.

Alleviation (59b75fbee1):

While allowance is properly consumed during the creation of a redemption request, allowance remains validated in the `As4626::cancelRedeemRequest`.

As a redemption request should be possible to cancel even if the original requester does not have any allowance anymore, we re-iterate our original advice to omit allowance checks and supplement it with a recommendation to apply the `opportunityCost` allowance adjustment opportunistically (i.e. if `currentAllowance - opportunityCost < 0`, allowance should be configured to `0`).

Alleviation (efbeab6478):

The code has been alleviated per our recommendation, omitting the allowance related checks during redemption request cancellations and ensuring that the allowance of the `_operator` is reduced up to `0` safely.

A62-08M: Inexistent Protection Against Re-Initialization

Type	Severity	Location
Logical Fault	Minor	As4626.sol:L46-L62

Description:

The `As4626::init` function does not prevent against re-initialization, causing the timestamps of the `last` data entry to be corrupted as well as permitting the name, symbol, and decimals of the `ERC20` representation of the contract to be adjusted post-deployment.

Impact:

A severity of minor has been assigned as the function is privileged, however, its impact is significant as fees can be lost and impersonation attacks can be performed.

Example:

```
src/abstract/As4626.sol
```

```
SOL

39  /**
40   * @dev Initializes the contract with the provided ERC20 metadata, core
addresses, and fees.
41   * Only the admin can call this function.
42   * @param _erc20Metadata The ERC20 metadata including name, symbol, and decimals.
43   * @param _coreAddresses The core addresses including the fee collector address.
44   * @param _fees The fees structure.
45   */
46 function init(
47     Erc20Metadata calldata _erc20Metadata,
48     CoreAddresses calldata _coreAddresses,
```

Example (Cont.):

SOL

```
49      Fees calldata _fees
50  ) public virtual onlyAdmin {
51      // check that the fees are not too high
52      setFees(_fees);
53      feeCollector = _coreAddresses.feeCollector;
54      req.redemptionLocktime = 6 hours;
55      last.accountedSharePrice = weiPerShare;
56      last.accountedProfit = weiPerShare;
57      last.feeCollection = uint64(block.timestamp);
58      last.liquidate = uint64(block.timestamp);
59      last.harvest = uint64(block.timestamp);
60      last.invest = uint64(block.timestamp);
61      ERC20._init(_erc20Metadata.name, _erc20Metadata.symbol,
62      _erc20Metadata.decimals);
62 }
```

Recommendation:

We advise the function to prevent re-invocation via a dedicated variable, ensuring the contract cannot be re-initialized.

Alleviation (59b75fbee1):

The Astrolab DAO team specified that they intend to supply an `initialized` public `bool` that will prevent re-initialization, however, no such change has been incorporated in the codebase yet.

As such, we consider this exhibit open in the codebase's current state.

Alleviation (efbeab6478):

An `if` clause was introduced ensuring that the `_initialized` flag of the `ERC20` parent implementation is `false` and reverting otherwise.

As such, we consider this exhibit properly alleviated.

A62-09M: Potentially Invalid Cancellation Assumption

Type	Severity	Location
Logical Fault	Minor	As4626.sol:L687, L688

Description:

The `As4626::cancelRedeemRequest` will update the `totalClaimableRedemption` data entry even if the redemption request has not been factored in a liquidation call as the `As4626::isRequestClaimable` function does not guarantee a liquidation has taken place.

Impact:

Redemption requests that should be able to be cancelled may result in an uncaught underflow due to an incorrect assumption in relation to whether a liquidation that satisfies the redemption has been performed or not.

Example:

```
src/abstract/As4626.sol
SOL
687 if (isRequestClaimable(request.timestamp) )
688     req.totalClaimableRedemption -= shares;
```

Recommendation:

We advise the code to ensure a liquidation has happened after the request has occurred, ensuring that the `totalClaimableRedemption` has a high likelihood of incorporating the shares that were meant to be liquidated.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The conditional has been updated to ensure that the redemption amount has been factored in a liquidation call by validating whether the request's timestamp is older than the last liquidation that occurred.

A62-10M: Improper Accounting of Fees in Downward Price Action

Type	Severity	Location
Logical Fault	Major	As4626.sol:L301, L304, L321

Description:

The `As4626::_collectFees` function will become permanently inaccessible if a downward price action has occurred while a non-zero `claimableAssetFees` value exists. Such an action will cause the `As4626::_collectFees` function to continue execution while all values yielded by the `AsAccounting::computeFees` function are `0`, resulting in the `accountedSharePrice` being configured to `0`.

This will cause any invocation of `AsAccounting::computeFees` to first compute a `change` equal to the share price itself (which is invalid), and then cause the code to yield a division-by-zero error due to attempting to calculate the `profit`.

Impact:

All fees will become permanently inaccessible if a downward action occurs for the share and at least a single withdrawal / deposit has occurred for the vault which is a highly likely scenario.

Example:

```
src/abstract/As4626.sol

SOL

293 /**
294 * @notice Trigger a fee collection: mints shares to the feeCollector
295 */
296 function _collectFees() internal nonReentrant returns (uint256 toMint) {
297
298     if (feeCollector == address(0))
299         revert AddressZero();
300
301     (uint256 assets, uint256 price, uint256 profit, uint256 feesAmount) =
302     AsAccounting.computeFees(IAs4626(address(this)));
303 }
```

Example (Cont.):

```
SOL

303     // sum up all fees: feesAmount (perf+mgmt) + claimableAssetFees (entry+exit)
304     toMint = convertToShares(feesAmount + claimableAssetFees, false);
305
306     // do not mint nor emit event if there are no fees to collect
307     if (toMint == 0)
308         return 0;
309
310     emit FeeCollection(
311         feeCollector,
312         assets,
313         price,
314         profit, // basis AsMaths.BP_BASIS**2
315         feesAmount,
316         toMint
317     );
318     _mint(feeCollector, toMint);
319     last.feeCollection = uint64(block.timestamp);
320     last.accountedAssets = assets;
321     last.accountedSharePrice = price;
322     last.accountedProfit = profit;
323     last.accountedSupply = totalSupply();
324     claimableAssetFees = 0;
325 }
```

Recommendation:

We advise either the `AsAccounting::computeFees` implementation to yield non-zero values with a zero `feesAmount` when returning early, or the `As4626::_collectFees` function to return early if just `feesAmount` is 0.

While we advise the former of the two to prevent time-based fees from accumulating when the share moves in a downward manner, different approaches can also be utilized such as waiting until the share price rebounds to the latest tracked one before charging fees.

Alleviation (59b75fbee1d8f3dee807c928f18be41c58b904e1):

The relevant function has been relocated to the `StrategyV5Agent` (`StrategyV5Agent::_collectFees`) contract and the relevant `AsAccounting` function has been renamed to `AsAccounting::claimableDynamicFees`.

In the renamed implementation, a case of no fees will properly yield the correct `price` as well as `assets` value, permitting the logic to function properly thus alleviating this exhibit in full.

A62-11M: Incorrect Implementation of EIP-7540

Type	Severity	Location
Logical Fault	Major	As4626.sol:L585-L590, L625

Description:

The `As4626` contract is meant to comply with the **EIP-7540** standard, however, it deviates from it in both its `interface` as well as the implementations of the various functions as denoted in the standard.

As an example, the `As4626::requestRedeem` function will invoke the

`IERC7540RedeemReceiver::onERC7540RedeemReceived` function on the `msg.sender` rather than the `_owner`.

Impact:

The `As4626` is not compatible with the **EIP-7540** standard, and one of the callbacks it performs during redemption requests is done so to the caller rather than the `_owner` which is invalid behaviour.

Example:

src/abstract/As4626.sol

SOL

```
579 /**
580  * @notice Initiate a redeem request for shares
581  * @param _shares Amount of shares to redeem
582  * @param _operator Address initiating the request
583  * @param _owner The owner of the shares to be redeemed
584 */
585 function requestRedeem(
586     uint256 _shares,
587     address _operator,
588     address _owner,
```

Example (Cont.):

```
SOL

589     bytes memory _data
590 ) public nonReentrant whenNotPaused returns (uint256 _requestId) {
591     if (_operator != msg.sender || (_owner != msg.sender && allowance(_owner,
592     _operator) < _shares))
593         revert Unauthorized();
594     if (_shares == 0 || balanceOf(_owner) < _shares)
595         revert AmountTooLow(_shares);
596
597     Erc7540Request storage request = req.byOwner[_owner];
598     if (request.operator != _operator) request.operator = _operator;
599
600     last.sharePrice = sharePrice();
601     if (request.shares > 0) {
602         if (request.shares > _shares)
603             revert AmountTooLow(_shares);
604
605         // reinit the request (re-added lower)
606         req.totalRedemption -= AsMaths.min(
607             req.totalRedemption,
608             request.shares
609         );
610         // compute request vwap
611         request.sharePrice =
612             ((last.sharePrice * (_shares - request.shares)) + (request.sharePrice
613             * request.shares)) /
614             _shares;
615     } else {
616         request.sharePrice = last.sharePrice;
617     }
618 }
```

Example (Cont.):

```
SOL

617     _requestId = ++requestId;
618     request.requestId = _requestId;
619     request.shares = _shares;
620     request.timestamp = block.timestamp;
621     req.totalRedemption += _shares;
622
623     if(_data.length != 0) {
624         // the caller contract must return the bytes4 value "0x0102fde4"
625         if(IERC7540RedeemReceiver(msg.sender).onERC7540RedeemReceived(_operator,
626             _owner, _requestId, _data) != 0x0102fde4)
627             revert Unauthorized();
628
629     }
630     emit RedeemRequest(_owner, _operator, _owner, _shares);
631 }
```

Recommendation:

We advise either the code to be substantially updated to comply with the **EIP-7540** standard, or to remove support of **EIP-7540** and instead implement a custom **EIP-7540** adaptation removing unnecessary traits such as the `IERC7540RedeemReceiver` callback.

We consider either of the two approaches as valid alleviations to this exhibit given that the **EIP-7540** is not yet mature.

Alleviation (59b75fbee1):

The code was updated to accommodate for **EIP-7540**, however, the standard itself underwent an update in between the preliminary report and its revision.

As an example, the `IERC7540::requestRedeem` function definition denotes a `receiver` argument on which the callback should be performed on instead of the `_owner`.

EIP-7540 integration should be revised based on the latest implementation of the standard as of 04-15-2024, and as an extension to the aforementioned recommendation we advise the

`As4626::requestDeposit` concept to be revised as it implements a dangerous polyfill in which the `_receiver` will expect state mutations as described in the **EIP-7540** and **this chapter in particular**.

Alleviation (efbeab6478):

The contract was refactored to achieve **EIP-7540** compliancy solely in relation to redemption requests due to size limitations the contract must abide by and the updates involved in the **EIP-7540** standard itself.

We observed that **EIP-7540** compliancy is still not achieved in two significant areas that pertain to redemption requests.

The first area of concern is the `As4626::claimableRedeemRequest` function and the fact that it does not align with the relevant **EIP-7540** implementation as [described here](#). As such, integrators will be unable to reliably invoke the `As4626::claimableRedeemRequest` function to assess the portion of funds that are claimable for a particular `_owner`.

The other area of concern is the `As4626::requestRedeem` function implementation itself, and the fact that it overwrites the previous `_shares` requested rather than incrementing them. **Per the standard itself:**

Assumes control of `shares` from `owner` and submits a Request for asynchronous `redeem`. This places the Request in Pending state, with a corresponding increase in `pendingRedeemRequest` for the amount `shares`.

As the present mechanism will overwrite the previous request if done for the same receiver, it will not behave per the standard and thus break compliancy.

We advise both deviancies to be alleviated so as to ensure the contract is and remains **EIP-7540** compliant.

As an additional comment, the `As4626::totalPendingWithdrawRequest` function is mistyped and should be corrected.

Alleviation (cf5194da53):

The Astrolab DAO team evaluated our follow-up review of the exhibit and proceeded with addressing the three concerns raised within it.

Specifically, a `As4626::claimableRedeemRequest` polyfill was introduced that complies with the **EIP-7540** function signature, the `As4626::totalPendingWithdrawRequest` typographic mistake was corrected, and the `As4626::requestRedeem` function was refactored to treat the input `_shares` as an increment of the existing redeem request if it exists.

As all **EIP-7540** related compatibility concerns have been addressed by the Astrolab DAO team, we consider this exhibit fully alleviated.

A62-12M: Inexistent Reservation of Shares

Type	Severity	Location
Logical Fault	Major	As4626.sol:L591, L593, L619

Description:

The `As4626 : : requestRedeem` function will permit a user to request a redemption to be fulfilled at a later date. This request will reserve a portion of the available funds in the strategy and will cause a liquidation to occur to satisfy it.

The flaw in the current implementation is that a redemption request does not reserve the underlying **EIP-20** balance, enabling a user to create multiple redemption requests with the same fungible **EIP-20** balance across multiple accounts.

Impact:

It is possible to cause the strategy to no longer operate by creating multiple redemption requests that must be honoured by the system's liquidation mechanisms.

Example:

```
src/abstract/As4626.sol

SOL

579 /**
580 * @notice Initiate a redeem request for shares
581 * @param _shares Amount of shares to redeem
582 * @param _operator Address initiating the request
583 * @param _owner The owner of the shares to be redeemed
584 */
585 function requestRedeem(
586     uint256 _shares,
587     address _operator,
588     address _owner,
```

Example (Cont.):

```
SOL

589     bytes memory _data
590 ) public nonReentrant whenNotPaused returns (uint256 _requestId) {
591     if (_operator != msg.sender || (_owner != msg.sender && allowance(_owner,
592     _operator) < _shares))
593         revert Unauthorized();
594     if (_shares == 0 || balanceOf(_owner) < _shares)
595         revert AmountTooLow(_shares);
596
597     Erc7540Request storage request = req.byOwner[_owner];
598     if (request.operator != _operator) request.operator = _operator;
599
600     last.sharePrice = sharePrice();
601     if (request.shares > 0) {
602         if (request.shares > _shares)
603             revert AmountTooLow(_shares);
604
605         // reinit the request (re-added lower)
606         req.totalRedemption -= AsMaths.min(
607             req.totalRedemption,
608             request.shares
609         );
610         // compute request vwap
611         request.sharePrice =
612             ((last.sharePrice * (_shares - request.shares)) + (request.sharePrice
613             * request.shares)) /
614             _shares;
615     } else {
616         request.sharePrice = last.sharePrice;
617     }
618 }
```

Example (Cont.):

```
SOL

617     _requestId = ++requestId;
618     request.requestId = _requestId;
619     request.shares = _shares;
620     request.timestamp = block.timestamp;
621     req.totalRedemption += _shares;
622
623     if(_data.length != 0) {
624         // the caller contract must return the bytes4 value "0x0102fde4"
625         if(IERC7540RedeemReceiver(msg.sender).onERC7540RedeemReceived(_operator,
626             _owner, _requestId, _data) != 0x0102fde4)
627             revert Unauthorized();
628     }
629     emit RedeemRequest(_owner, _operator, _owner, _shares);
630 }
```

Recommendation:

We advise the `As4626::requestRedeem` function to ensure that the `_shares` being submitted as part of the request are correctly locked and to prevent their transfer or usage until the request is either cancelled or fulfilled.

Alleviation (59b75fbbe1):

The code was updated to overload the `ERC20::transfer` function, however, the `ERC20::transferFrom` function continues to permit shares meant for a request to be transferred.

Alleviation (efbeab6478):

The `ERC20::transferFrom` function has been overridden as well, ensuring that all **EIP-20** transfer related functions correctly impose the pending redemption request amount limitation.

As4626Abstract Manual Review Findings

AAT-01M: EIP-7540 Incompatibility

Type	Severity	Location
Standard Conformity	Minor	As4626Abstract.sol:L40-L45

Description:

The `RedeemRequest` event declaration within the `As4626Abstract` contract does not comply with the specification of the EIP.

Impact:

A severity of minor has been assigned as the incompatibility will solely affect off-chain consumers of these events, however, it is imperative that the incompatibility is rectified.

Example:

```
src/abstract/As4626Abstract.sol
```

```
SOL
```

```
39 // ERC7540
40 event RedeemRequest(
41     address indexed sender,
42     address indexed operator,
43     address indexed owner,
44     uint256 assets
45 );
```

Recommendation:

We advise the event arguments as well as names to be updated to comply with the **EIP-7540** standard as otherwise compatibility with it should not be advertised.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The event has been updated to become fully compliant with the latest **EIP-7540** standard definition as of 04-15-2024, alleviating this exhibit.

AsArrays Manual Review Findings

AAS-01M: Incorrect EVM Memory Assumptions

Type	Severity	Location
Logical Fault	Major	AsArrays.sol:L80-L89, L96-L106, L112-L121

Description:

The `AsArrays::ref`, `AsArrays::unref`, and thereby `AsArrays::slice` (for the `uint256[]` data type) are invalid implementations as they do not conform to the intricacies of the EVM's memory space.

Specifically, the `AsArrays::slice` function incorrectly assumes that an array pointer can be transformed to a new array by shifting the pointer's value in `32` byte increments per the elements the user wishes to skip. This is incorrect, as a valid array pointer will contain the array's length in the first `32` byte slot and the elements after it, meaning that shifting the pointer will result in an array that has a length equal to the `begin - 1` element.

In turn, this will cause the `data := ptr` assignment in `AsArrays::unref` to produce an array with length equal to `self[begin - 1]` containing all entries that fit within that length as well as corrupt memory data due to an "overflow" of the allocated memory as a result of the overwritten array length should it exceed the initial size.

On the other hand, the `AsArrays::unref` function will instantiate a `data` pointer with an array with a specified `size`, however, the `size` will be overwritten by the ensuing assignment. This means that whatever the expected `size`, the resulting `data` array will use the aforementioned `self[begin - 1]` entry (or the actual size, if `begin` is `0`) as the `length` and the local declaration will be immediately discarded.

As a final note, the `AsArrays::testRefUnref` function is an ineffective test as it will not mutate the length of the array nor will it skip any elements in which case the malfunctions we described do not surface.

Impact:

Any `AsArrays::slice` operation will either result in corrupted data or transaction failure, either of which can be considered of significant severity.

Example:

src/libs/AsArrays.sol

SOL

```
80  /**
81   * @notice Returns a reference to the array
82   * @param data array to be referenced
83   * @return ptr reference of the array
84   */
85 function ref(uint256[] memory data) internal pure returns (uint ptr) {
86     assembly {
87       ptr := data
88     }
89 }
```

Example (Cont.):

```
SOL

90
91  /**
92   * @notice Returns dereferenced array (slice) starting at ptr and containing
93   * size elements
94   * @param ptr reference of the array
95   * @return array of given size
96   */
97  function unref(uint256 ptr, uint256 size) internal pure returns (uint256[] memory) {
98      uint256[] memory data = new uint256[](size);
99      assembly {
100          data := ptr
101          // safer:
102          // for { let i := 0 } lt(i, size) { i := add(i, 1) } {
103          //     mstore(add(data, add(0x20, mul(i, 0x20))), mload(add(ptr, mul(i,
104          // 0x20))))
105          // }
106      }
107
108 /**
109  * @notice Used to test memory pointers on the current evm
110  * @return true - memory ok, false - memory error
111  */
112 function testRefUnref() internal pure returns (bool) {
113     uint256[] memory dt = new uint256[](3);
114     for (uint i = 0; i < dt.length; i++) {
115         dt[i] = i;
116     }
117     uint256 wptr = ref(dt);
```

Example (Cont.):

```
SOL

118     uint256[] memory data;
119     data = unref(wptr, 3);
120     return data.length == 3 && data[0] == 0 && data[1] == 1 && data[2] == 2;
121 }
122
123 /**
124 * @notice Returns a slice of the array
125 * @param self Storage array containing uint256 type variables
126 * @param begin Index of the first element to include in the slice
127 * @param end Index of the last element to include in the slice
128 * @return slice of the array
129 */
130 function slice(uint256[] memory self, uint256 begin, uint256 end) internal pure
returns (uint256[] memory) {
131     require(begin < end && end <= self.length);
132     return unref(ref(self) + begin * 0x20, end - begin);
133 }
```

Recommendation:

We advise the overall approach to efficient array slicing to be revised as it is presently incorrect, causes data corruptions as well as potential unhandled errors.

To note, slices of in-memory arrays can already be acquired using the slice syntax (i.e. `arr[a:b]`) for `calldata` arrays and could be a viable replacement to a custom `slice` implementation.

Alleviation (59b75fbee1):

The codebase was refactored with the `AsArrays::unref` implementation removed and the `AsArrays::slice` implementation revised, however, the `AsArrays::slice` implementation remains incorrect.

Specifically, it will copy less than the actual elements it is meant to due to calculating the `end` pointer as `add(src, length)` instead of `add(src, mul(length, 0x20))` thus causing the ensuing `for` loop to terminate early.

In turn, this will lead to the array incorrectly yielding zeroed out entries instead of failing. We advise the `end` pointer to be updated properly so as to fully alleviate this exhibit.

As a final note, both `AsArrays::slice` implementations incorrectly calculate the `end` pointer until which the iteration should run.

Alleviation (efbeab6478):

The Astrolab DAO team evaluated the follow-up alleviation chapter of this exhibit and opted to omit the functions from the contract entirely, rendering this exhibit alleviated by omission.

AAS-02M: Incorrect Usage of Memory

Type	Severity	Location
Language Specific	Major	AsArrays.sol:L24, L43, L64

Description:

The referenced statements will write to the `0x60` memory pointer which is meant to represent the initial value of dynamic memory arrays, thereby corrupting all future array instantiations.

Impact:

Any invocation of the `AsArrays::sum`, `AsArrays::max`, or `AsArrays::min` functions will cause future array instantiations to be corrupted.

Example:

```
src/libs/AsArrays.sol
SOL
36 /**
37  * @notice Returns the max value in an array.
38  * @param self Storage array containing uint256 type variables
39  * @return value The highest value in the array
40 */
41 function max(uint256[] storage self) public view returns (uint256 value) {
42     assembly {
43         mstore(0x60, self.slot)
44         value := sload(keccak256(0x60, 0x20))
45     }
}
```

Example (Cont.):

```
SOL
46      for {
47          let i := 0
48      } lt(i, sload(self.slot)) {
49          i := add(i, 1)
50      }
51          switch gt(sload(add(keccak256(0x60, 0x20), i)), value)
52          case 1 {
53              value := sload(add(keccak256(0x60, 0x20), i))
54          }
55      }
56  }
```

Recommendation:

We advise the code to utilize the `self.slot` directly or to properly reserve memory using the free memory pointer at `0x80`.

Alleviation (59b75fbe1):

The code was updated to load the free memory pointer at `0x40`, however, the actual free memory pointer is not updated after its memory has been utilized which is incorrect.

For more details on how to securely utilize the free memory pointer, kindly consult the relevant **Solidity documentation resource**.

Alleviation (efbeab6478):

The free memory pointer is updated correctly in all relevant instances, and is updated twice redundantly in the `AsArrays::sum` function.

As the original issue has been alleviated properly, we consider this exhibit addressed despite the inefficiency described.

AsCast Manual Review Findings

ACT-01M: Potentially Insecure Address Cast

Type	Severity	Location
Input Sanitization	Minor	AsCast.sol:L119-L121

Description:

The `AsCast::toAddress` function will cast the input `bytes32` variable to an `address` without validating that the variable does not have any corrupt bits.

Impact:

Dirty bits in the `bytes32` variable will not affect the end-result of the casting operation but may affect other contextual assumptions in the caller of the function.

Example:

```
src/libs/AsCast.sol
SOL
114 /**
115 * @dev Converts a bytes32 value to an address.
116 * @param b The bytes32 value to convert.
117 * @return The converted address.
118 */
119 function toAddress(bytes32 b) internal pure returns (address) {
120     return address(uint160(uint256(b)));
121 }
```

Recommendation:

We advise the code to cast the `uint256` representation of the `bytes32` variable to a `uint160` variable safely (i.e. via `AsCast::toUInt160`), ensuring that there are no dirty bits in the representation cast.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The code was updated to invoke the `AsCast::toUInt160` function as advised, ensuring that all `address` casts are safely performed.

AsManageable Manual Review Findings

AME-01M: Invalid Conditional Evaluation

Type	Severity	Location
Logical Fault	Informational	AsManageable.sol:L86-L96, L172

Description:

The `AsManageable::checkRoleAcceptance` function will return early if the role being accepted is the `KEEPER_ROLE`, however, the `pendingAcceptance` entry will never have the `KEEPER_ROLE` as a `role` due to the fact that the `AsManageable::grantRole` will configure it solely when the `role` is either the `DEFAULT_ADMIN_ROLE` or the `MANAGER_ROLE`.

Example:

```
src/abstract/AsManageable.sol
```

```
SOL

70  /**
71   * @notice Grant a role to an account
72   *
73   * @dev If the role is admin, the account will have to accept the role
74   * The acceptance period will expire after TIMELOCK_PERIOD has passed
75   */
76 function grantRole(
77     bytes32 role,
78     address account
79 )
```

Example (Cont.):

```
SOL

80     public
81     override
82     onlyRole(getRoleAdmin(role))
83     {
84         require(!hasRole(role, account));
85
86         if (role == DEFAULT_ADMIN_ROLE || role == MANAGER_ROLE) {
87
88             pendingAcceptance[account] = PendingAcceptance({
89                 // only get replaced if admin, managers can coexist
90                 replacing: role == DEFAULT_ADMIN_ROLE
91                     ? msg.sender
92                     : address(0),
93                 timestamp: block.timestamp,
94                 role: role
95             });
96         } else {
97             _grantRole(role, account);
98         }
99     }
```

Recommendation:

We advise the code to be updated, potentially by adjusting the `if` conditional of the `AsManageable::grantRole` function to also execute the `pendingAcceptance` path when the input `role` is the `KEEPER_ROLE`.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The Astrolab DAO team indicated that they will address this point with an acceptable remediation, however, it remains open in the latest version of the codebase and specifically the

`AccessController::acceptRole` function that implements the original contract's purpose.

As the exhibit does not pose a security concern, we will consider it acknowledged but advise the Astrolab DAO team to potentially revisit it.

AME-02M: Detachment of Authorized Role

Type	Severity	Location
Logical Fault	Major	AsManageable.sol:L152, L158, L160

Description:

The `AsManageable::acceptRole` function will permit the caller to accept any role they wish regardless of what was initially authorized to them via the `AsManageable::grantRole` function as the input `role` argument is utilized instead of the `acceptance.role` data entry.

Impact:

It is presently possible to acquire a different role than the one you have been authorized for (i.e. acquire the `DEFAULT_ADMIN_ROLE` while authorized for the `MANAGER_ROLE`) as well as cause the deletion of the `DEFAULT_ADMIN_ROLE` by accepting such an authorization whilst granting a different role.

Example:

```
src/abstract/AsManageable.sol
```

```
SOL

145 /**
146 * @notice Accept an admin role and revoke the old admin
147 *
148 * @dev If the role is admin or manager, the account will have to accept the role
149 * The acceptance will expire after TIMELOCK_PERIOD + VALIDITY_PERIOD has passed
150 * Old admin will be revoked and new admin will be granted
151 */
152 function acceptRole(bytes32 role) external {
153     PendingAcceptance memory acceptance = pendingAcceptance[msg.sender];
154 }
```

Example (Cont.):

```
SOL

155     _checkRoleAcceptance(acceptance);
156     if (acceptance.replacing != address(0)) {
157         // if replacing, revoke the old role
158         _revokeRole(acceptance.role, acceptance.replacing);
159     }
160     _grantRole(role, msg.sender);
161     delete pendingAcceptance[msg.sender];
162 }
```

Recommendation:

We advise the code to remove the `role` argument entirely, and to utilize the `pendingAcceptance` payload for all the data it requires.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

Role acceptance is properly validated in the `AccessController::acceptRole` function and specifically the `AccessController::checkRoleAcceptance` validation mechanism which has replaced the original `AsManageable` implementation.

AsMaths Manual Review Findings

AMS-01M: Improper Absolute Function Implementation

Type	Severity	Location
Mathematical Operations	Informational	AsMaths.sol:L248

Description:

The `AsMaths::abs` function is expected to yield the absolute value of the input `int256` number in its `uint256` representation, however, in doing so the function will not properly handle the value `type(int256).min` even though it is representable by the `uint256` the conversion occurs to.

This is due to the fact that all signed integers have one less value in the positive range as a result of the bit signifying the polarity of the number.

Impact:

As the code would simply `revert` instead of yielding a corrupt value, we consider its severity to be informational.

Example:

```
src/libs/AsMaths.sol
SOL
242 /**
243 * @notice Get the absolute value of a signed integer
244 * @param x The input signed integer
245 * @return The absolute value of the input
246 */
247 function abs(int256 x) internal pure returns (uint256) {
248     return uint256(x > 0 ? x : -x);
249 }
```

Recommendation:

We advise a conditional to be introduced, ensuring that `uint256(type(int256).max) + 1` is yielded if the input `x` is equal to the `type(int256).min` value.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The case of `x` being equivalent to `type(int256).min` is now adequately handled by the `AsMaths::abs` function, ensuring that the value is calculated safely for all possible inputs.

AsProxy Manual Review Findings

APY-01M: Reservation of Function Signatures

Type	Severity	Location
Standard Conformity	 Informational	AsProxy.sol:L57, L65, L73

Description:

Any `Proxy` implementation is meant to relay calls to its logic contract and should not implement any functions of its own to avoid function signature clashes (i.e. a function signature being present in both the `Proxy` and its logic implementation).

In such cases, the function signature in the `Proxy` implementation will take precedence preventing the function from the logic contract from ever being invoked via it.

Impact:

The probability of a function signature collision is low but not unlikely given that only 4 bytes are utilized of the resulting function's hash. As such, it is advised that these implementations are instead present in the logic contract to ensure that the proxy is a pass-through contract rather than one with logic within it.

Example:

```
src/abstract/AsProxy.sol

SOL

54 /**
55  * @notice Returns the proxy initialization state
56 */
57 function initialized() public view virtual returns (bool) {
58     return _implementation() != address(0);
59 }
60
61 /**
62  * @dev Returns the EIP-897 address of the implementation contract
63  * @return The address of the implementation contract
```

Example (Cont.):

```
SOL

64     */
65     function implementation() external view virtual returns (address) {
66         return _implementation();
67     }
68
69     /**
70      * @dev Returns the EIP-897 proxy type
71      * @return The proxy type
72     */
73     function proxyType() external pure virtual returns (uint256) {
74         return 2;
75     }
```

Recommendation:

We advise the functions to be implemented by the logic implementation instead, ensuring that all function signatures are properly forwarded to the logic contract.

Alleviation (59b75fbee1d8f3dee807c928f18be41c58b904e1):

The `AsProxy` implementation was removed from the codebase after consideration of the audit report's outputs and its usage has been replaced by vanilla `delegatecall` integrations.

As such, all exhibits relevant to it have been marked as no longer applicable.

APY-02M: Potentially Insecure Utilization of Scratch Space

Type	Severity	Location
Language Specific	Minor	AsProxy.sol:L29, L31

Description:

The `AsProxy::_delegateWithSignature` function attempts to mimic the `Proxy::_delegate` implementation by taking control over the full memory scratch space whose security relies entirely on the way the function is invoked as well as the primitives that are used around its invocation.

Impact:

A severity of minor has been assigned due to the fact that the top-level call that leads to the `AsProxy::_delegateWithSignature` function's execution has been confirmed as being the final statement in each code block.

In spite of this, we still advise proper memory reservation to occur as it represents a somewhat small gas increase while significantly bolstering the security of these relayed calls.

Example:

```
src/abstract/AsProxy.sol
SOL
17  /**
18   * @notice Delegate a call to an implementation contract using a function
19   * @param _implementation The address of the implementation contract
20   * @param _signature The function signature to delegate
21   */
22 function _delegateWithSignature(
23     address _implementation,
24     string memory _signature
25 ) internal {
26     bytes4 selector = bytes4(keccak256(bytes(_signature)));

```

Example (Cont.):

```
SOL

27     assembly {
28         // Store selector at the beginning of the calldata
29         mstore(0x0, selector)
30         // Copy the rest of calldata (skipping the first 4 bytes of the original
function signature)
31         calldatacopy(0x4, 0x4, sub(calldatasize(), 0x4))
32         let result := delegatecall(
33             gas(),
34             _implementation,
35             0x0,
36             calldatasize(),
37             0,
38             0
39         )
40         let size := returndatasize()
41         let ptr := mload(0x40)
42         returndatacopy(ptr, 0, size)
43
44         switch result
45         case 0 {
46             revert(ptr, size)
47         }
48         default {
49             return(ptr, size)
50         }
51     }
52 }
```

Recommendation:

We strongly advise against utilizing the scratch space based on the fact that the

`AsProxy::_delegateWithSignature` function is invoked within other functions, the usage of `keccak256` prior to the `assembly` block which utilizes **the scratch space itself**, and the fact that the memory required by the function is dynamic and reliant on the call-data of the top-level call.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The `AsProxy` implementation was removed from the codebase after consideration of the audit report's outputs and its usage has been replaced by vanilla `delegatecall` integrations.

As such, all exhibits relevant to it have been marked as no longer applicable.

APY-03M: Insecure Forwarded Payload

Type	Severity	Location
Logical Fault	Major	AsProxy.sol:L31

Description:

The `AsProxy::_delegateWithSignature` function will forward the payload attached to the transaction's `calldata` to the `selector` associated with the input function `_signature` of the function.

Based on the way the `AsProxy::_delegateWithSignature` function is invoked within the codebase, the relayed payload will be outright incorrect or contain superfluous data points in the following cases:

Impact:

Any `StrategyV5Chainlink::updateAsset` / `StrategyV5Pyth::updateAsset` call will result in misbehaviour as it will relay an improper `_priceFactor` which we consider a significant misbehaviour.

Example:

src/abstract/AsProxy.sol

SOL

Example (Cont.):

```
SOL

27     assembly {
28         // Store selector at the beginning of the calldata
29         mstore(0x0, selector)
30         // Copy the rest of calldata (skipping the first 4 bytes of the original
function signature)
31         calldatacopy(0x4, 0x4, sub(calldatasize(), 0x4))
32         let result := delegatecall(
33             gas(),
34             _implementation,
35             0x0,
36             calldatasize(),
37             0,
38             0
39         )
40         let size := returndatasize()
41         let ptr := mload(0x40)
42         returndatacopy(ptr, 0, size)
43
44         switch result
45         case 0 {
46             revert(ptr, size)
47         }
48         default {
49             return(ptr, size)
50         }
51     }
52 }
```

Recommendation:

The flaw arises from the fact that the transaction's `calldata` is utilized, and the `calldata` remains the same regardless of how many `internal` functions are invoked as only an external call can mutate the `calldata`.

As the function is never used to actually forward a dynamic `calldata` based payload, we advise a `bytes memory` argument to be introduced to the function that is in turn forwarded, ensuring that the data the `_implementation` contract receives is accurate and expectable.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The `AsProxy` implementation was removed from the codebase after consideration of the audit report's outputs and its usage has been replaced by vanilla `delegatecall` integrations.

As such, all exhibits relevant to it have been marked as no longer applicable.

AsSequentialSet Manual Review Findings

ASS-01M: Improper Sequential Set Shift Operation

Type	Severity	Location
Logical Fault	Informational	AsSequentialSet.sol:L79-L87

Description:

The `AsSequentialSet::shift` operation will break the sequential nature of the set as it will replace the first element with the last element of the set and then pop the first element from the end of the array, thereby breaking its order.

Example:

src/libs/AsSequentialSet.sol

```
SOL

75  /**
76   * @dev Removes the first element from the sequential set.
77   * @param q The sequential set.
78   */
79 function shift(Set storage q) internal {
80     if (q.data.length == 0) {
81         revert EmptySet();
82     }
83     delete q.index[q.data[0]];
84     q.data[0] = q.data[q.data.length - 1];
```

Example (Cont.):

SOL

```
85     q.index[q.data[0]] = 1;  
86     q.data.pop();  
87 }
```

Recommendation:

We advise this trait to be re-evaluated, as the set is no longer sequential via these operations.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The Astrolab DAO team evaluated this exhibit and clarified that the "Sequential" keyword is meant to refer to memory allocation rather than how the elements are ordered. The team proceeded to rename the library as `AsIterableSet` to better reflect this fact, addressing any confusion that the exhibit arose from.

ASS-02M: Inexistent Prevention of Duplicate Elements

Type	Severity	Location
Logical Fault	Minor	AsSequentialSet.sol:L37, L94, L111

Description:

The `AsSequentialSet` is inherently incompatible with duplicate entries due to its index system and would cause a fatal corruption of the dataset if any such entry is added.

Impact:

The `AsSequentialSet` as presently utilized will prevent this misbehaviour from manifesting, however, it is crucial that the duplicate entry limitation is enforced at the `library` level to avoid this behaviour surfacing as part of future development efforts.

Example:

```
src/libs/AsSequentialSet.sol

SOL

32  /**
33   * @dev Adds an element to the end of the sequential set.
34   * @param q The sequential set.
35   * @param o The element to be added.
36   */
37 function push(Set storage q, bytes32 o) internal {
38     q.data.push(o);
39     q.index[o] = uint32(q.data.length);
40 }
```

Recommendation:

We advise the code to prevent duplicate entries by ensuring that the `q.index` of an entry being added is `0`.

Alleviation (59b75fbee1):

While the `AsIterableSet::push` and `AsIterableSet::insert` functions have been updated to prevent duplicates, the `AsIterableSet::unshift` function continues to permit them rendering this exhibit partially alleviated.

Alleviation (efbeab6478):

A `require` check was introduced at the top of the `AsIterableSet::unshift` function that disallows duplicate entries correctly, rendering this exhibit fully alleviated.

ASS-03M: Invalid Sequential Set Shift Operation

Type	Severity	Location
Logical Fault	Major	AsSequentialSet.sol:L83-L86

Description:

If the `length` of the set is `1`, the `AsSequentialSet::shift` operation will retain a non-zero `q.index` for the entry being removed even though it is no longer present in the array.

Impact:

When the last element of the array is shifted, the element will have a non-zero `index` even though it is no longer present in the set which is invalid and would cause `AsSequentialSet::has` evaluations to yield `true` after other elements are placed as well as incorrect behaviour if anyone attempts to remove it.

Example:

src/libs/AsSequentialSet.sol

SOL

```
75  /**
76   * @dev Removes the first element from the sequential set.
77   * @param q The sequential set.
78   */
79 function shift(Set storage q) internal {
80     if (q.data.length == 0) {
81         revert EmptySet();
82     }
83     delete q.index[q.data[0]];
84     q.data[0] = q.data[q.data.length - 1];
```

Example (Cont.):

SOL

```
85     q.index[q.data[0]] = 1;  
86     q.data.pop();  
87 }
```

Recommendation:

We advise the code to instead **delete** the `q.index` of the last remaining element and simply **pop** it if the `q.data.length` value is `1`, ensuring that the entries are correctly updated.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

Our recommendation was adhered to, deleting the index of the last element (i.e. the only one in the array `s.data[0]`) and issuing a **pop** operation to the `s.data` array.

ASS-04M: Invalid Sequential Set Unshift Operation

Type	Severity	Location
Logical Fault	● Major	AsSequentialSet.sol:L98-L100

Description:

The `AsSequentialSet::unshift` function will overwrite the last element of the array if the `q.data.length` is non-zero and will also not update its index, corrupting the sequential set.

Impact:

Whenever an element is unshifted and the array is not empty, the last entry of the set will be removed from the system while its index will yield a non-zero entry thereby causing `AsSequentialSet::has` evaluations to yield `true` as well as incorrect behaviour if anyone attempts to remove it.

Example:

src/libs/AsSequentialSet.sol

SOL

```
89  /**
90   * @dev Adds an element to the beginning of the sequential set.
91   * @param q The sequential set.
92   * @param o The element to be added.
93   */
94 function unshift(Set storage q, bytes32 o) internal {
95     if (q.data.length == 0) {
96       q.data.push(o);
97     } else {
98       q.data[q.data.length - 1] = q.data[0];
```

Example (Cont.):

SOL

```
99         q.index[q.data[0]] = uint32(q.data.length);  
100        q.data[0] = o;  
101    }  
102    q.index[o] = 1;  
103 }
```

Recommendation:

We advise the code to instead `push` to the `q.data` array and perform the referenced statements afterwards, ensuring that no data is overwritten from the sequential set and that all indexes are correct.

To note, this would also break the order of the sequential set as specified in a separate exhibit and an alternative approach should be utilized if the order is expected to remain the same.

Alleviation (59b75fbee1):

While the code was refactored to `push` and perform the relevant index updates, the `s.index[0]` assignment of `1` was relocated within the `else` block of the `AsIterableSet::unshift` function which causes an `AsIterableSet::unshift` operation on an empty `s.data` structure to not update the `index` of the element added.

We advise the `s.index[0]` update to be relocated outside the `if-else` block as it was in the original implementation, ensuring that the `index` of the unshifted `o` element is correctly maintained under all scenarios.

Alleviation (efbeab6478):

The `q.index` assignment has been relocated outside the `if-else` clause per the original implementation, addressing this exhibit in full.

StrategyV5 Manual Review Findings

SV5-01M: Implementation & Documentation Mismatch

Type	Severity	Location
Logical Fault	Informational	StrategyV5.sol:L498

Description:

The inline documentation of the referenced statement denotes that:

only invest 90% of liquidity for buffered flows

However, the full `As4626Abstract::available` amount is utilized for the investment preview.

Impact:

The system is presently inefficient as no liquidation buffer is utilized and the documentation does not match the implementation of the code.

Example:

src/abstract/StrategyV5.sol

```
SOL

489 /**
490  * @dev Preview the amounts that would be invested based on the given amount
491  * @param _amount Amount of asset to invest with
492  * @return amounts uint256[8] Previewed investment amounts for each input in
asset
493 */
494 function previewInvest(
495     uint256 _amount
496 ) public view returns (uint256[8] memory amounts) {
497     if (_amount == 0)
498         _amount = available(); // only invest 90% of liquidity for buffered flows
```

Example (Cont.):

```
SOL

499     int256[8] memory excessInput = _excessInputLiquidity(invested() + _amount);
500     for (uint8 i = 0; i < inputLength; i++) {
501         if (_amount < 10) break; // no leftover
502         if (excessInput[i] < 0) {
503             uint256 need = _inputToAsset(excessInput[i].abs(), i);
504             if (need > _amount)
505                 need = _amount;
506             amounts[i] = need;
507             _amount -= need;
508         }
509     }
510 }
```

Recommendation:

We advise the code to properly utilize only 90% of the `As4626Abstract::available` amount to ensure that a buffer is permitted for potential liquidations that may occur.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The Astrolab DAO team evaluated this exhibit and identified that it represented a simple discrepancy between the latest code implementation and the in-line documentation that accompanies it.

Specifically, the "90%" threshold can be imposed via `inputWeights` rendering a flat reduction unnecessary and in reality inefficient in the latest system.

As such, we consider this exhibit as alleviated and have downgraded its severity to reflect a documentational discrepancy.

SV5-02M: Discrepancy of Liquidation Preview

Type	Severity	Location
Mathematical Operations	Minor	StrategyV5.sol:L471, L474, L476

Description:

The `StrategyV5::previewLiquidate` function will **add** the minimum between `totalPendingAssetRequest() + allocated.bp(150)` and `allocated` to the input `_amount`, however, the ensuing subtraction will fail if the `_amount` that results exceeds the `allocated` value.

Impact:

As the `StrategyV5::previewLiquidate` function is solely utilized by off-chain software, the impact of this flaw would solely translate to off-chain services and whether they handle revert errors of the `StrategyV5::previewLiquidate` function or not.

Example:

src/abstract/StrategyV5.sol

```
SOL

465 /**
466  * @dev Preview the amounts that would be liquidated based on the given amount
467  * @param _amount Amount of asset to liquidate with (0 ==
totalPendingAssetRequest() + allocated.bp(100))
468  * @return amounts uint256[8] Previewed liquidation amounts for each input
469  */
470 function previewLiquidate(
471     uint256 _amount
472 ) public view returns (uint256[8] memory amounts) {
473     uint256 allocated = invested();
474     _amount += AsMaths.min(totalPendingAssetRequest() + allocated.bp(150),
allocated); // defaults to requests + 1% offset to buffer flows
```

Example (Cont.):

SOL

```
475     // excessInput accounts for the weights and the cash available in the
476     strategy
477     int256[8] memory excessInput = _excessInputLiquidity(allocated - _amount);
478     for (uint8 i = 0; i < inputLength; i++) {
479         if (_amount < 10) break; // no leftover
480         if (excessInput[i] > 0) {
481             uint256 need = _inputToAsset(excessInput[i].abs(), i);
482             if (need > _amount)
483                 need = _amount;
484             amounts[i] = _assetToInput(need, i);
485             _amount -= need;
486         }
487     }
```

Recommendation:

We advise the `totalPendingAssetRequest() + allocated.bp(150)` value to be added to the `_amount` directly, and the `_amount` to be consequently assigned to the minimum between the calculated value and the value of `allocated` ensuring that a subtraction overflow cannot occur.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

Our recommendation has been applied to the letter, incrementing the `_amount` by the relevant buffer (updated to 0.5% from 1.5% in the latest implementation) and then calculating the minimum between the new `_amount` and the `allocated` value.

SV5-03M: Insecure Casting Operations

Type	Severity	Location
Mathematical Operations	Minor	StrategyV5.sol:L419, L420, L448, L449

Description:

The referenced operations will cast a `uint256` variable to its signed representation (`int256`) without proper bound checks.

Impact:

Any cast-based overflow operation will not be properly detected by the Solidity version utilized, potentially causing misbehaviours in the calculations referenced if the cast values manage to exceed the maximum of an `int256`.

Example:

```
src/abstract/StrategyV5.sol
```

```
SOL
```

```
407 /**
408  * @dev Calculate the excess weight for a given input index
409  * @param _index Index of the input
410  * @param _total Total invested amount
411  * @return int256 Excess weight (/AsMaths.BP_BASIS)
412 */
413 function _excessWeight(
414     uint8 _index,
415     uint256 _total
416 ) internal view returns (int256) {
```

Example (Cont.):

SOL

```
417     if (_total == 0) _total = invested();
418     return
419     int256(invested(_index).mulDiv(AsMaths.BP_BASIS, _total)) -
420     int256(uint256(inputWeights[_index]));
421 }
```

Recommendation:

We advise each cast to be performed safely, ensuring the value being cast is less-than the maximum supported by the `int256` data type (i.e. `type(int256).max`).

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The Astrolab DAO team indicated that they plan to enforce safety checks for the relevant casting operations in a future iteration of the codebase per the linked GitHub discussion.

As such, we consider this exhibit to be safely acknowledged.

StrategyV5Agent Manual Review Findings

SVA-01M: Discrepant Allowance Maintenance

Type	Severity	Location
Logical Fault	Minor	StrategyV5Agent.sol:L53, L143-L144

Description:

The `StrategyV5Agent::setSwapperAllowance` function will ensure that a new swapper will be properly authorized to swap the reward tokens of the contract, however, the `StrategyV5Agent::setRewardTokens` function fails to do this thereby causing newly configured reward tokens to lack the necessary approval to be utilized.

Impact:

Configuration of new reward tokens will cause them to be inoperable by the swapper and would require multiple actions for the swapper to be properly approved for them.

Example:

```
src/abstract/StrategyV5Agent.sol
SOL
134 /**
135 * @notice Sets the reward tokens
136 * @param _rewardTokens array of reward tokens
137 */
138 function setRewardTokens(
139     address[] calldata _rewardTokens
140 ) public onlyManager {
141     if (_rewardTokens.length > 8) revert Unauthorized();
142     for (uint8 i = 0; i < _rewardTokens.length; i++) {
143         rewardTokens[i] = _rewardTokens[i];
```

Example (Cont.):

SOL

```
144     rewardTokenIndex[_rewardTokens[i]] = i+1;
145 }
146 rewardLength = uint8(_rewardTokens.length);
147 }
```

Recommendation:

We advise the `StrategyV5Agent::setRewardTokens` function to properly set allowances, ensuring the `swapper` can utilize them as necessary.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The `StrategyV5Agent::_setRewardTokens` function, representing an internalization of the original `StrategyV5Agent::setRewardTokens` logic, was updated to properly supply approvals for the newly configured reward tokens rendering this exhibit alleviated.

SVA-02M: Improper No-Op Logic Statement

Type	Severity	Location
Language Specific	Minor	StrategyV5Agent.sol:L49

Description:

The referenced statement will not result in any functional change to the code as it will evaluate a ternary operator and not utilize the result.

Impact:

The code is meant to treat a `0` value allowance as the maximum but presently ignores it.

Example:

```
src/abstract/StrategyV5Agent.sol

SOL

41  /**
42   * @notice Sets the swapper allowance
43   * @param _amount Amount of allowance to set
44   */
45 function setSwapperAllowance(uint256 _amount) public onlyAdmin {
46     address swapperAddress = address(swapper);
47     // we keep the possibility to set allowance to 0 in case of a change of
swapper
48     // default is to approve MAX_UINT256
49     _amount != 0 ? _amount : MAX_UINT256;
50 }
```

Example (Cont.):

SOL

```
51     for (uint256 i = 0; i < rewardLength; i++) {
52         if (rewardTokens[i] == address(0)) break;
53         IERC20Metadata(rewardTokens[i]).approve(swapperAddress, _amount);
54     }
55     for (uint256 i = 0; i < inputLength; i++) {
56         if (address(inputs[i]) == address(0)) break;
57         inputs[i].approve(swapperAddress, _amount);
58     }
59     asset.approve(swapperAddress, _amount);
60 }
```

Recommendation:

We advise the behaviour of the `StrategyV5Agent::setSwapperAllowance` to be validated and the ternary operator to either be removed or incorporated within it.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The code was updated to properly utilize the result of the ternary statement in an assignment to the `_amount` variable, addressing this exhibit.

SVA-03M: Inexistent Erasure of Previous Approvals

Type	Severity	Location
Logical Fault	Minor	StrategyV5Agent.sol:L111, L126, L143

Description:

The various functions of the `StrategyV5Agent` contract that permit the inputs, reward tokens, and underlying asset to be adjusted do not erase the previously present approval to the swapper, permitting lingering approvals to remain in the code.

Impact:

Approvals that are a result of replaced assets will remain to the `swapper` even if it is replaced, signifying a potential flaw in the system that can affect funds.

Example:

```
src/abstract/StrategyV5Agent.sol
```

```
SOL

114 /**
115  * @notice Sets the input tokens (strategy internals), make sure to liquidate()
them first
116  * @param _inputs array of input tokens
117  * @param _weights array of input weights
118  */
119 function setInputs(
120     address[] calldata _inputs,
121     uint16[] calldata _weights
122 ) public onlyAdmin {
123     if (_inputs.length > 8) revert Unauthorized();
```

Example (Cont.):

SOL

```
124     address swapperAddress = address(swapper);
125     for (uint8 i = 0; i < _inputs.length; i++) {
126         inputs[i] = IERC20Metadata(_inputs[i]);
127         inputDecimals[i] = inputs[i].decimals();
128         inputWeights[i] = _weights[i];
129         inputs[i].approve(swapperAddress, MAX_UINT256);
130     }
131     inputLength = uint8(_inputs.length);
132 }
133
134 /**
135  * @notice Sets the reward tokens
136  * @param _rewardTokens array of reward tokens
137 */
138 function setRewardTokens(
139     address[] calldata _rewardTokens
140 ) public onlyManager {
141     if (_rewardTokens.length > 8) revert Unauthorized();
142     for (uint8 i = 0; i < _rewardTokens.length; i++) {
143         rewardTokens[i] = _rewardTokens[i];
144         rewardTokenIndex[_rewardTokens[i]] = i+1;
145     }
146     rewardLength = uint8(_rewardTokens.length);
147 }
```

Recommendation:

We advise the code to properly erase any approval that previously existed, ensuring that no lingering approvals to potentially unauthorized swappers remain.

To note, the input and reward token configuration functions will also need to iterate up to the end of the `inputs` / `rewardTokens` array respectively to ensure a shrink of the array will also cause approvals to be erased.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The `StrategyV5Agent::_setInputs` and `strategyV5Agent::_setRewardTokens` functions, both representing internalized implementations of their original `public` un-prefixed counterparts, have been updated to erase any previously existing approval when tokens are updated effectively alleviating this exhibit in full.

SVA-04M: Inexistent Protection Against Re-Initialization

Type	Severity	Location
Logical Fault	Minor	StrategyV5Agent.sol:L31-L39

Description:

The `StrategyV5Agent::init` function does not prevent against re-initialization, permitting the asset to be updated without the proper flow defined in `StrategyV5Agent::updateAsset`.

Impact:

A severity of minor has been assigned as the function is privileged, however, its impact is significant as the asset's immediate adjustment without a proper migration can cause the strategy to misbehave greatly.

Example:

```
src/abstract/StrategyV5Agent.sol
SOL
27 /**
28  * @notice Initialize the strategy
29  * @param _params StrategyBaseParams struct containing strategy parameters
30  */
31 function init(StrategyBaseParams calldata _params) public onlyAdmin {
32     // setInputs(_params.inputs, _params.inputWeights);
33     setRewardTokens(_params.rewardTokens);
34     asset = IERC20Metadata(_params.coreAddresses.asset);
35     assetDecimals = asset.decimals();
36     weiPerAsset = 10**assetDecimals;
```

Example (Cont.):

SOL

```
37     updateSwapper(_params.coreAddresses.swapper);
38     As4626.init(_params.erc20Metadata, _params.coreAddresses, _params.fees);
39 }
```

Recommendation:

We advise the function to prevent re-invocation via a dedicated variable, ensuring the contract cannot be re-initialized.

Alleviation (59b75fbee1):

The Astrolab DAO team specified that they intend to supply an `initialized` public `bool` that will prevent re-initialization, however, no such change has been incorporated in the codebase yet.

As such, we consider this exhibit open in the codebase's current state.

Alleviation (efbeab6478):

Initialization protection has been introduced to the `As4626::__init` function, rendering this exhibit alleviated as a result.

SVA-05M: Insecure Approval Operations

Type	Severity	Location
Logical Fault	Minor	StrategyV5Agent.sol:L53, L57, L111, L129

Description:

The referenced approval operations may fail if the underlying token prevents approval reconfigurations when a non-zero approval exists.

Impact:

Presently, a reconfiguration of the inputs of the `StrategyV5Agent` may fail due to one of the tokens being present in both the old and new inputs and thus causing the approval to fail.

Example:

```
src/abstract/StrategyV5Agent.sol
```

```
SOL

114 /**
115  * @notice Sets the input tokens (strategy internals), make sure to liquidate()
them first
116  * @param _inputs array of input tokens
117  * @param _weights array of input weights
118 */
119 function setInputs(
120     address[] calldata _inputs,
121     uint16[] calldata _weights
122 ) public onlyAdmin {
123     if (_inputs.length > 8) revert Unauthorized();
```

Example (Cont.):

SOL

```
124     address swapperAddress = address(swapper);
125     for (uint8 i = 0; i < _inputs.length; i++) {
126         inputs[i] = IERC20Metadata(_inputs[i]);
127         inputDecimals[i] = inputs[i].decimals();
128         inputWeights[i] = _weights[i];
129         inputs[i].approve(swapperAddress, MAX_UINT256);
130     }
131     inputLength = uint8(_inputs.length);
132 }
```

Recommendation:

We advise usage of OpenZeppelin's `SafeERC20` library and specifically its `SafeERC20::forceApprove` function, ensuring that approval overwrites are correctly performed.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

All `IERC20::approve` instances have been replaced by OpenZeppelin's `SafeERC20::forceApprove` function, ensuring that they will be performed properly regardless of the underlying allowance's state thereby alleviating this exhibit.

StrategyV5Chainlink Manual Review Findings

SVC-01M: Inexistent Prevention of Data Corruption

Type	Severity	Location
Input Sanitization	Minor	StrategyV5Chainlink.sol:L55-L59

Description:

The `StrategyV5Chainlink::setPriceFeed` function does not ensure that no previous entry exists for either the `_address` or `_feed`, allowing corruption of their respective data entries in the system.

Impact:

A severity of minor has been assigned due to the function's privileged nature.

Example:

```
src/abstract/StrategyV5Chainlink.sol
SOL

49  /**
50   * @dev Sets the validity duration for a single price feed
51   * @param _address The address of the token we want the feed for
52   * @param _feed The pricefeed address for the token
53   * @param _validity The new validity duration in seconds
54   */
55 function setPriceFeed(address _address, IChainlinkAggregatorV3 _feed, uint256
56   _validity) public onlyAdmin {
57     feedByAsset[_address] = _feed;
58     decimalsByFeed[_feed] = feedByAsset[_address].decimals();
59     validityByFeed[feedByAsset[_address]] = _validity;
```

Example (Cont.):

SOL

59 }

Recommendation:

We advise the `StrategyV5Chainlink::setPriceFeed` function to ensure that the `feedByAsset[_address]` entry is zero, and to utilize a different variable to track whether the `_feed` has been configured to be validated as such.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The Astrolab DAO team evaluated this exhibit and has opted to purposefully not validate whether a pre-existing feed exists in the relocated `ChainlinkProvider::_setFeed` function as they wish to be able to set temporary "identity" feeds for bridged assets when proper feeds are not present.

As such, we consider this exhibit alleviated **based on the fact that the Astrolab DAO team will responsibly employ the data feed configurations.**

SVC-02M: Inexistent Validation of Prices

Type	Severity	Location
Logical Fault	Minor	StrategyV5Chainlink.sol:L178, L199

Description:

In direct contradiction with the `ChainlinkUtils::getPriceUsd` function, the referenced Chainlink queries do not ensure the yielded `price` is positive.

Impact:

The likelihood of a Chainlink oracle misbehaving is considered low, however, validation of the yielded `price` should always be performed as a fail-safe.

Example:

```
src/abstract/StrategyV5Chainlink.sol
```

```
SOL
```

```
173 function _usdToInput(
174     uint256 _amount,
175     uint8 _index
176 ) internal view returns (uint256) {
177     IChainlinkAggregatorV3 feed = feedByAsset[address(inputs[_index])];
178     (, int256 price, , uint256 updateTime, ) = feed.latestRoundData();
179     if (block.timestamp > (updateTime + validityByFeed[feed]))
180         revert InvalidOrStaleValue(updateTime, price);
181     return
182     _amount.mulDiv(
```

Example (Cont.):

SOL

```
183         10 ** (uint256(decimalsByFeed[feed]) + inputDecimals[_index] - 6),
184         uint256(price)
185     ); // eg. (1e6+1e8+1e6)-(1e8+1e6) = 1e6
186 }
```

Recommendation:

We advise such validation to be introduced, preventing invalid prices from being consumed as acceptable by the system.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The `StrategyV5Chainlink` implementation has been superseded by the `ChainlinkProvider` implementation, and the relevant statement is now located in the `ChainlinkProvider::_toUsdBp` function.

A validity check for the reported price has been introduced in the relocated code, properly alleviating this exhibit.

As4626 Code Style Findings

A62-01C: Inefficient `mapping` Lookups

Type	Severity	Location
Gas Optimization	● Informational	As4626.sol:L166, L182

Description:

The linked statements perform key-based lookup operations on `mapping` declarations from storage multiple times for the same key redundantly.

Example:

src/abstract/As4626.sol

SOL

```
166 Erc7540Request storage request = req.byOwner[_owner];
167 uint256 claimable = claimableRedeemRequest(_owner);
168 last.sharePrice = sharePrice();
169
170 uint256 price = (claimable >= _shares)
171     ? AsMaths.min(last.sharePrice, request.sharePrice) // worst of if pre-
existing request
172     : last.sharePrice; // current price
173
174 // amount/shares cannot be higher than the share price (dictated by the inline
convertToAssets below)
175 if (_amount > _shares.mulDiv(price * weiPerAsset, weiPerShare ** 2))
```

Example (Cont.):

SOL

```
176     revert AmountTooHigh(_amount);
177
178 if (msg.sender != _owner)
179     _spendAllowance(_owner, msg.sender, _shares);
180
181 if (claimable >= _shares) {
182     req.byOwner[_owner].shares -= _shares;
183     req.totalRedemption -= AsMaths.min(_shares, req.totalRedemption); // min 0
184     req.totalClaimableRedemption -= AsMaths.min(
185         _shares,
186         req.totalClaimableRedemption
187     ); // min 0
188 } else {
```

Recommendation:

As the lookups internally perform an expensive `keccak256` operation, we advise the lookups to be cached wherever possible to a single local declaration that either holds the value of the `mapping` in case of primitive types or holds a `storage` pointer to the `struct` contained.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The optimization has been applied per our recommendation, using the `request` storage pointer that already exists for the `shares` member mutation in the second highlighted line.

A62-02C: Redundant Duplication of Code

Type	Severity	Location
Code Style	Informational	As4626.sol:L146

Description:

The referenced statement will locally perform the statements of the `As4626: :previewDeposit` function redundantly.

Example:

```
src/abstract/As4626.sol
```

```
SOL
```

```
146 shares = _deposit(_amount, convertToShares(_amount,  
false).subBp(exemptionList[_receiver] ? 0 : fees.entry), _receiver);
```

Recommendation:

We advise the function to be invoked directly, optimizing the legibility of the code.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The referenced statements are no longer part of the `As4626::safeDeposit` function, rendering this exhibit no longer applicable.

A62-03C: Redundant Parenthesis Statements

Type	Severity	Location
Code Style	Informational	As4626.sol:L170, L801

Description:

The referenced statements are redundantly wrapped in parenthesis' (())).

Example:

```
src/abstract/As4626.sol
```

```
SOL
```

```
170 uint256 price = (claimable >= _shares)
```

Recommendation:

We advise them to be safely omitted, increasing the legibility of the codebase.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The first of the two referenced instances is no longer applicable whilst the second instance has been corrected in its updated form rendering this exhibit fully addressed.

A62-04C: Repetitive Value Literal

Type	Severity	Location
Code Style	Informational	As4626.sol:L96, L175, L190, L200

Description:

The linked value literal is repeated across the codebase multiple times.

Example:

```
src/abstract/As4626.sol
```

```
SOL
```

```
96 if (_amount > maxDeposit(address(0)) || _shares > _amount.mulDiv(weiPerShare ** 2,  
last.sharePrice * weiPerAsset))
```

Recommendation:

We advise it to be set to a `constant` variable instead optimizing the legibility of the codebase.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

All referenced instances of `weiPerShare ** 2` have been replaced by a `_WEI_PER_SHARE_SQUARED` constant per our recommendation, optimizing the code's legibility.

As4626Abstract Code Style Findings

AAT-01C: Generic Typographic Mistakes

Type	Severity	Location
Code Style	Informational	As4626Abstract.sol:L61, L63, L66, L69, L70, L71, L75, L82, L83, L86

Description:

The referenced lines contain typographical mistakes (i.e. `private` variable without an underscore prefix) or generic documentational errors (i.e. copy-paste) that should be corrected.

Example:

```
src/abstract/As4626Abstract.sol
```

```
SOL
```

```
61 uint256 internal constant MAX_UINT256 = type(uint256).max;
```

Recommendation:

We advise them to be corrected enhancing the legibility of the codebase.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

All referenced declarations have been appropriately prefixed with an underscore when necessary, addressing this exhibit in full.

AAT-02C: Improper Declaration of Abstract Function

Type	Severity	Location
Standard Conformity	Informational	As4626Abstract.sol:L107

Description:

The `As4626Abstract::invested` function is meant to be `virtual` and implemented by derivative implementations, however, an empty declaration is present that would permit it to be invoked and yield `0` if it is not overridden.

Example:

```
src/abstract/As4626Abstract.sol
```

```
SOL
```

```
102 /**
103  * @notice Total amount of inputs denominated in asset
104  * @dev Abstract function to be implemented by the strategy
105  * @return Amount of assets
106 */
107 function invested() public view virtual returns (uint256) {}
```

Recommendation:

We advise the function to be declared without a code block (`{}`) to ensure it is overridden by derivative implementations.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The referenced function was adjusted in visibility and renamed to `As4626Abstract::_invested`, incorporating our recommendation by no longer specifying an empty code block.

AsAccessControl Code Style Findings

AAC-01C: Inefficient Usage of Utility Functions

Type	Severity	Location
Gas Optimization	Informational	AsAccessControl.sol:L126, L137, L149

Description:

In the referenced instances, the `_roles[role]` lookup will be redundantly performed multiple times due to using utility functions that also fetch its storage location.

Example:

```
src/abstract/AsAccessControl.sol
```

```
SOL
```

```
143 /**
144  * @dev Internal function to revoke a role from an account.
145  * @param role The role to revoke.
146  * @param account The account to revoke the role from.
147 */
148 function _revokeRole(bytes32 role, address account) internal virtual {
149     if (hasRole(role, account)) {
150         _roles[role].members.remove(account.toBytes32());
151         emit RoleRevoked(role, account, msg.sender);
152     }
```

Example (Cont.):

SOL

153 }

Recommendation:

We advise the function invocations to be replaced by their statements directly, caching the result of `_roles[role]` to a local `RoleState storage` variable that can be re-used and thus optimize the gas cost of the functions.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The `[AsAccessControl]` implementation has been sunset by the standalone `AccessController` implementation which incorporates a significant portion of the original code.

The ported implementations of `AccessController::_setRoleAdmin`, `AccessController::_grantRole`, and `[AccessController::_revokeRole]` properly incorporate the optimization outlined by no longer utilizing utility functions.

As a result, we consider this exhibit alleviated in the implementation that supersedes the original.

AAC-02C: Redundant Input Argument

Type	Severity	Location
Gas Optimization	Informational	AsAccessControl.sol:L106

Description:

The `AsAccessControl::renounceRole` function will accept an input argument that will always be mandated as equal to the `msg.sender`.

Example:

src/abstract/AsAccessControl.sol

```
SOL

101 /**
102  * @dev Renounce a role for the sender account.
103  * @param role The role to renounce.
104  * @param account The account renouncing the role.
105 */
106 function renounceRole(bytes32 role, address account) external virtual {
107     if (account != msg.sender) revert Unauthorized();
108     _revokeRole(role, account);
109 }
```

Recommendation:

We advise the referenced input argument to be omitted, ensuring that a role renunciation only requires the role that is being renounced.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The `AsAccessControl` implementation has been sunset by the standalone `AccessController` implementation which incorporates a significant portion of the original code.

The ported implementation of `AccessController::renounceRole` incorporates our recommendation to omit the input argument and replacing it with direct use of the `msg.sender`.

As a result, we consider this exhibit alleviated in the implementation that supersedes the original.

AAC-03C: Redundant Local Variable

Type	Severity	Location
Gas Optimization	Informational	AsAccessControl.sol:L126

Description:

The referenced statement will declare a `previousAdminRole` local variable that is solely utilized once within the code block.

Example:

src/abstract/AsAccessControl.sol

```
SOL

120 /**
121 * @dev Internal function to set the admin role for a given role.
122 * @param role The role to set the admin role of.
123 * @param adminRole The admin role to be set.
124 */
125 function _setRoleAdmin(bytes32 role, bytes32 adminRole) internal virtual {
126     bytes32 previousAdminRole = getRoleAdmin(role);
127     _roles[role].adminRole = adminRole;
128     emit RoleAdminChanged(role, previousAdminRole, adminRole);
129 }
```

Example (Cont.):

```
SOL

130
131 /**
132 * @dev Internal function to grant a role to an account.
133 * @param role The role to grant.
134 * @param account The account to grant the role to.
135 */
136 function _grantRole(bytes32 role, address account) internal virtual {
137     if (!hasRole(role, account)) {
138         _roles[role].members.push(account.toBytes32());
139         emit RoleGranted(role, account, msg.sender);
```

Recommendation:

We advise the `AsAccessControl::getRoleAdmin` evaluation to be directly utilized as input to the `RoleAdminChanged` event, and the event's emission to be relocated prior to the `_roles` data entry's adjustment.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The `AsAccessControl` implementation has been sunset by the standalone `AccessController` implementation which incorporates a significant portion of the original code.

The ported implementation of `AccessController::_setRoleAdmin` properly applies our recommended optimization by emitting the `RoleAdminChanged` event before mutating the `role.adminRole` data entry.

As a result, we consider this exhibit alleviated in the implementation that supersedes the original.

AsAccounting Code Style Findings

AAG-01C: Repetitive Value Literal

Type	Severity	Location
Code Style	● Informational	AsAccounting.sol:L126, L127, L128

Description:

The linked value literal is repeated across the codebase multiple times.

Example:

```
src/libs/AsAccounting.sol
```

```
SOL
```

```
126 _fees.entry <= 200 && // 2%
```

Recommendation:

We advise it to be set to a `constant` variable instead optimizing the legibility of the codebase.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

Proper `constant` declarations have been introduced for all relevant maximum fee limitations inclusive of the ones referenced by this exhibit, thereby addressing it in full.

AsArrays Code Style Findings

AAS-01C: Ineffectual Usage of Safe Arithmetics

Type	Severity	Location
Language Specific	Informational	AsArrays.sol:L149

Description:

The linked mathematical operation is guaranteed to be performed safely by surrounding conditionals evaluated in either `require` checks or `if-else` constructs.

Example:

```
src/libs/AsArrays.sol
SOL
146 require(begin < end && end <= self.length);
147
148 // Calculate the number of elements in the slice
149 uint256 sliceLength = end - begin;
```

Recommendation:

Given that safe arithmetics are toggled on by default in `pragma` versions of `0.8.x`, we advise the linked statement to be wrapped in an `unchecked` code block thereby optimizing its execution cost.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The referenced statement is no longer present in the codebase in any shape or form rendering this exhibit inapplicable.

AAS-02C: Inefficient Iteration of Search Loops

Type	Severity	Location
Gas Optimization	Informational	AsArrays.sol:L47, L68

Description:

The referenced loops will iterate from `0` to identify the maximum and minimum respectively, however, their `value` entries are already initialized with the first entry of the array.

Example:

src/libs/AsArrays.sol

```
SOL

36  /**
37   * @notice Returns the max value in an array.
38   * @param self Storage array containing uint256 type variables
39   * @return value The highest value in the array
40  */
41 function max(uint256[] storage self) public view returns (uint256 value) {
42     assembly {
43         mstore(0x60, self.slot)
44         value := sload(keccak256(0x60, 0x20))
45     }
```

Example (Cont.):

```
SOL

46      for {
47          let i := 0
48          } lt(i, sload(self.slot)) {
49              i := add(i, 1)
50          } {
51              switch gt(sload(add(keccak256(0x60, 0x20), i)), value)
52              case 1 {
53                  value := sload(add(keccak256(0x60, 0x20), i))
54              }
55          }
56      }
57  }
58
59 /// @notice Returns the minimum value in an array.
60 /// @param self Storage array containing uint256 type variables
61 /// @return value The highest value in the array
62 function min(uint256[] storage self) public view returns (uint256 value) {
63     assembly {
64         mstore(0x60, self.slot)
65         value := sload(keccak256(0x60, 0x20))
66
67         for {
68             let i := 0
69             } lt(i, sload(self.slot)) {
70                 i := add(i, 1)
71             } {
72                 switch gt(sload(add(keccak256(0x60, 0x20), i)), value)
73                 case 0 {
```

Example (Cont.):

SOL

```
74             value := sload(add(keccak256(0x60, 0x20), i))  
75         }  
76     }  
77 }  
78 }
```

Recommendation:

We advise the loops to begin at `1`, optimizing each function's gas cost by one iteration.

Alleviation (cf5194da53ebf026da6c8efa74daada96719cc71):

Both loops will now begin iteration at the `1` index, optimizing their gas cost by one iteration.

AAS-03C: Inefficient Iterator Type

Type	Severity	Location
Gas Optimization	Informational	AsArrays.sol:L169, L173, L177

Description:

The referenced `for` loops utilize a `uint64` variable as an iterator which is inefficient.

Example:

```
src/libs/AsArrays.sol
```

```
SOL
```

```
169 arr = new uint8[](n); for (uint64 i = 0; i < n; i++) arr[i] = a;
```

Recommendation:

As the EVM is built to operate on 32-byte (256-bit) data types, we advise the iterator types to be bumped to `uint256`, optimizing their gas cost.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

All referenced iterators have been updated to the `uint256` data type, optimizing the codebase as advised.

AAS-04C: Inexistent Error Messages

Type	Severity	Location
Code Style	Informational	AsArrays.sol:L131, L146

Description:

The linked `require` checks have no error messages explicitly defined.

Example:

```
src/libs/AsArrays.sol
```

```
SOL
```

```
131 require(begin < end && end <= self.length);
```

Recommendation:

We advise each to be set so to increase the legibility of the codebase and aid in validating the `require` checks' conditions.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

In-line documentation was introduced to clarify what the error is. Given that the contracts of the Astrolab DAO codebase tread closely to the bytecode size limit, we consider this approach as an adequate alleviation.

AAS-05C: Loop Iterator Optimizations

Type	Severity	Location
Gas Optimization	Informational	AsArrays.sol:L114, L155, L169, L173, L177

Description:

The linked `for` loops increment / decrement their iterator "safely" due to Solidity's built - in safe arithmetics (post-0.8.x).

Example:

```
src/libs/AsArrays.sol
```

```
SOL
```

```
114 for (uint i = 0; i < dt.length; i++) {
```

Recommendation:

We advise the increment / decrement operations to be performed in an `unchecked` code block as the last statement within each `for` loop to optimize their execution cost.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The referenced loop iterator increment statements have been relocated at the end of each respective `for` loop's body and have been unwrapped in an `unchecked` code block, optimizing their gas cost.

AsManageable Code Style Findings

AME-01C: Generic Typographic Mistakes

Type	Severity	Location
Code Style	● Informational	AsManageable.sol:L30, L31

Description:

The referenced lines contain typographical mistakes (i.e. `private` variable without an underscore prefix) or generic documentational errors (i.e. copy-paste) that should be corrected.

Example:

```
src/abstract/AsManageable.sol
```

```
SOL
```

```
30 uint256 private constant TIMELOCK_PERIOD = 2 days;
```

Recommendation:

We advise them to be corrected enhancing the legibility of the codebase.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The referenced variables have been renamed and their visibility specifier has been adjusted to `public`, effectively addressing this exhibit as the names are now correctly **not prefixed** with an underscore.

AME-02C: Inexistent Error Message

Type	Severity	Location
Code Style	Informational	AsManageable.sol:L84

Description:

The linked `require` check has no error message explicitly defined.

Example:

```
src/abstract/AsManageable.sol
```

```
SOL
```

```
84 require(!hasRole(role, account));
```

Recommendation:

We advise one to be set so to increase the legibility of the codebase and aid in validating the `require` check's condition.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The `require` check remains without an explicit error message or in-line documentation justifying it in its relocated `AccessController::grantRole` location, rendering the exhibit acknowledged.

AME-03C: Redundant Parenthesis Statements

Type	Severity	Location
Code Style	● Informational	AsManageable.sol:L140, L173, L175

Description:

The referenced statements are redundantly wrapped in parenthesis' (())).

Example:

```
src/abstract/AsManageable.sol
SOL
140 if ((role == DEFAULT_ADMIN_ROLE) && account == msg.sender)
```

Recommendation:

We advise them to be safely omitted, increasing the legibility of the codebase.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The first of the three referenced redundant parenthesis statements is no longer present in the codebase whilst the latter two could be justified as a legibility increase, rendering this exhibit ultimately alleviated.

AsMaths Code Style Findings

AMS-01C: Generic Typographic Mistakes

Type	Severity	Location
Code Style	Informational	AsMaths.sol:L22, L23, L24

Description:

The referenced lines contain typographical mistakes (i.e. `private` variable without an underscore prefix) or generic documentational errors (i.e. copy-paste) that should be corrected.

Example:

```
src/libs/AsMaths.sol
```

```
SOL
```

```
22 uint256 internal constant BP_BASIS = 10_000; // 50% == 5_000 == 5e3
```

Recommendation:

We advise them to be corrected enhancing the legibility of the codebase.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The referenced variables remain without an underscore prefix despite their `internal` visibility specification, rendering this exhibit acknowledged.

AMS-02C: Ineffectual Usage of Safe Arithmetics

Type	Severity	Location
Language Specific	Informational	AsMaths.sol:L149

Description:

The linked mathematical operation is guaranteed to be performed safely by surrounding conditionals evaluated in either `require` checks or `if-else` constructs.

Example:

```
src/libs/AsMaths.sol
```

```
SOL
```

```
149 return a > b ? a - b : b - a;
```

Recommendation:

Given that safe arithmetics are toggled on by default in `pragma` versions of `0.8.x`, we advise the linked statement to be wrapped in an `unchecked` code block thereby optimizing its execution cost.

Alleviation (cf5194da53ebf026da6c8efa74daada96719cc71):

The referenced subtractions have been wrapped in an `unchecked` code block, optimizing their gas cost.

AMS-03C: Inexistent Error Messages

Type	Severity	Location
Code Style	Informational	AsMaths.sol:L514, L862

Description:

The linked `require` checks have no error messages explicitly defined.

Example:

```
src/libs/AsMaths.sol
```

```
SOL
```

```
514 require(denominator > prod1);
```

Recommendation:

We advise each to be set so to increase the legibility of the codebase and aid in validating the `require` checks' conditions.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

While the former of the two `require` checks is accompanied by descriptive in-line documentation, the latter is not thus rendering this exhibit acknowledged.

AMS-04C: Redundant Parenthesis Statements

Type	Severity	Location
Code Style	Informational	AsMaths.sol:L129, L160, L257, L266, L270, L274, L278, L282

Description:

The referenced statements are redundantly wrapped in parenthesis' (())).

Example:

```
src/libs/AsMaths.sol
```

```
SOL
```

```
129 return (diff(a, b) <= val);
```

Recommendation:

We advise them to be safely omitted, increasing the legibility of the codebase.

Alleviation (cf5194da53ebf026da6c8efa74daada96719cc71):

The redundant parenthesis in the referenced statements have been safely omitted.

AsProxy Code Style Findings

APY-01C: Inefficient Generation of Selector

Type	Severity	Location
Gas Optimization	Informational	AsProxy.sol:L26

Description:

The `AsProxy::_delegateWithSignature` function will calculate the function selector locally from an input `string` which is significantly inefficient.

Example:

src/abstract/AsProxy.sol

```
SOL

17  /**
18   * @notice Delegate a call to an implementation contract using a function
19   * @param _implementation The address of the implementation contract
20   * @param _signature The function signature to delegate
21   */
22 function _delegateWithSignature(
23     address _implementation,
24     string memory _signature
25 ) internal {
26     bytes4 selector = bytes4(keccak256(bytes(_signature)));

```

Recommendation:

Given that the function signatures invoked are known at compile-time, we advise `interface` declarations for them to be utilized and specifically the `selector` syntax.

The `AsProxy::delegateWithSignature` function is invoked with the `StrategyV5Agent::init`, `StrategyV5Agent::updateAsset`, and `StrategyV5Abstract::setInputs` functions all of which can become part of an `interface` (i.e. `IStrategyV5Agent`) and accessed as advised (i.e. `IStrategyV5Agent.init.selector`).

Alleviation (59b75fbee1d8f3dee807c928f18be41c58b904e1):

The `AsProxy` implementation was removed from the codebase after consideration of the audit report's outputs and its usage has been replaced by vanilla `delegatecall` integrations.

As such, all exhibits relevant to it have been marked as no longer applicable.

AsRescuable Code Style Findings

ARE-01C: Improper Declarations of Abstract Functions

Type	Severity	Location
Standard Conformity	 Informational	AsRescuable.sol:L65, L98

Description:

The `AsRescuable::requestRescue` and `AsRescuable::rescue` functions are meant to be `virtual` and implemented by derivative implementations, however, an empty declaration is present in both that would permit each to be invoked.

Example:

```
src/abstract/AsRescuable.sol
SOL
64 // to be overridden with the proper access control by inheriting contracts
65 function requestRescue(address _token) external virtual {}
```

Recommendation:

We advise the functions to be declared without a code block (`{ }{ }`) to ensure they are overridden by derivative implementations.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

Both functions are now fully implemented by the `AsRescuable` implementation directly, rendering this exhibit no longer applicable.

ARE-02C: Inefficient Erasure of Request

Type	Severity	Location
Gas Optimization	Informational	AsRescuable.sol:L83-L84, L92-L93

Description:

The `AsRescuable::_rescue` function will erase the `rescueRequests` entry after the token has been transferred but will mark its `timestamp` as `0` before to prevent re-entrancies.

Example:

src/abstract/AsRescuable.sol

```
SOL

67  /**
68   * @dev Internal function to rescue tokens or native tokens (ETH) from the
69   * contract.
70   * @param _token The address of the token to be rescued. Use address(1) for
71   * native tokens (ETH).
72   * @notice This function can only be called by the receiver specified in the
73   * rescue request.
74   * @notice The rescue request must be initiated before the rescue timelock
75   * expires.
76   * @notice The rescue request remains valid until the rescue validity period
77   * expires.
78   * @notice If the rescue request is valid, the specified amount of tokens will be
79   * transferred to the receiver.
80   * @notice If the rescue request is not valid, a new rescue request will be set
81   * with the caller as the receiver.
82   * @notice Emits a Rescue event when the rescue is successful.
83   * @notice Emits a Rescue event when a new rescue request is set.
```

Example (Cont.):

```
SOL

77   */
78   function _rescue(address _token) internal {
79     RescueRequest storage req = rescueRequests[_token];
80     // check if rescue is pending
81     require(_isRescueUnlocked(req));
82
83     // reset timestamp to prevent reentrancy
84     rescueRequests[_token].timestamp = 0;
85
86     // send to receiver
87     if (_token == address(1)) {
88       payable(req.receiver).transfer(address(this).balance);
89     } else {
90       IERC20Metadata(_token).safeTransfer(req.receiver,
91       IERC20Metadata(_token).balanceOf(address(this)));
92     }
93     // reset pending request
94     delete rescueRequests[_token];
95   }
96 }
```

Recommendation:

As the `rescueRequests` entry is not utilized beyond the `AsRescuable::isRescueUnlocked` validation, we advise the entry to be deleted immediately after validation, optimizing the code's gas cost.

Alleviation (cf5194da53ebf026da6c8efa74daada96719cc71):

The inefficiency has been addressed by issuing the `delete` operation in place of the `timestamp` erasure statement, optimizing the code's gas cost.

ARE-03C: Inefficient `mapping` Lookups

Type	Severity	Location
Gas Optimization	● Informational	AsRescuable.sol:L79, L84

Description:

The linked statements perform key-based lookup operations on `mapping` declarations from storage multiple times for the same key redundantly.

Example:

```
src/abstract/AsRescuable.sol
```

```
SOL
```

```
79 RescueRequest storage req = rescueRequests[_token];
80 // check if rescue is pending
81 require(_isRescueUnlocked(req));
82
83 // reset timestamp to prevent reentrancy
84 rescueRequests[_token].timestamp = 0;
```

Recommendation:

As the lookups internally perform an expensive `keccak256` operation, we advise the lookups to be cached wherever possible to a single local declaration that either holds the value of the `mapping` in case of primitive types or holds a `storage` pointer to the `struct` contained.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The second highlighted instance properly utilizes the existing `req` storage pointer, optimizing the code as advised.

ARE-04C: Inexistent Error Messages

Type	Severity	Location
Code Style	Informational	AsRescuable.sol:L57, L81

Description:

The linked `require` checks have no error messages explicitly defined.

Example:

```
src/abstract/AsRescuable.sol
SOL
57 require(!_isRescueUnlocked(req));
```

Recommendation:

We advise each to be set so to increase the legibility of the codebase and aid in validating the `require` checks' conditions.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

While the latter of the two `require` checks is accompanied by descriptive in-line documentation, the former is not thus rendering this exhibit acknowledged.

AsRescuableAbstract Code Style Findings

ARA-01C: Optimization of Data Structure

Type	Severity	Location
Gas Optimization	Informational	AsRescuableAbstract.sol:L15

Description:

The `RescueRequest` data structure will occupy two storage slots redundantly as the `timestamp` value will fit reasonably within `96` bits as a Unix timestamp.

Example:

```
src/abstract/AsRescuableAbstract.sol
SOL
14 struct RescueRequest {
15     uint256 timestamp;
16     address receiver;
17 }
18 mapping(address => RescueRequest) internal rescueRequests;
```

Recommendation:

We advise the data type of the `timestamp` to be updated, ensuring that each `rescueRequests` entry occupies a single storage slot.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The `RescueRequest` data structure, now relocated to the `AsRescuable` implementation, has not applied the `timestamp` related optimization rendering this exhibit acknowledged.

AsSequentialSet Code Style Findings

ASS-01C: Ineffectual Usage of Safe Arithmetics

Type	Severity	Location
Language Specific	● Informational	AsSequentialSet.sol:L114, L145

Description:

The linked mathematical operations are guaranteed to be performed safely by surrounding conditionals evaluated in either `require` checks or `if-else` constructs.

Example:

```
src/libs/AsSequentialSet.sol
```

```
SOL
```

```
144 require(i > 0, "Element not found");
145 removeAt(q, i - 1);
```

Recommendation:

Given that safe arithmetics are toggled on by default in `pragma` versions of `0.8.x`, we advise the linked statements to be wrapped in `unchecked` code blocks thereby optimizing their execution cost.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The referenced arithmetic operations in their relocated `AsIterableSet` location are still performed using checked arithmetic, rendering this exhibit acknowledged.

ASS-02C: Inefficient Loop Limit Evaluations

Type	Severity	Location
Gas Optimization	Informational	AsSequentialSet.sol:L114, L250

Description:

The linked `for` loops evaluate their limit inefficiently on each iteration.

Example:

```
src/libs/AsSequentialSet.sol
```

```
SOL
```

```
114 for (uint256 j = q.data.length; j > i; j--) {
```

Recommendation:

We advise the statements within the `for` loop limits to be relocated outside to a local variable declaration that is consequently utilized for the evaluations to significantly reduce the codebase's gas cost. We should note the same optimization is applicable for storage reads present in those limits as they are newly read on each iteration (i.e. `length` members of arrays in storage).

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The former of the two loops is no longer present in the codebase whilst the latter remains unoptimized, rendering this exhibit acknowledged.

ASS-03C: Inexistent Error Message

Type	Severity	Location
Code Style	Informational	AsSequentialSet.sol:L173

Description:

The linked `require` check has no error message explicitly defined.

Example:

```
src/libs/AsSequentialSet.sol
```

```
SOL
```

```
173 require(i < q.data.length);
```

Recommendation:

We advise one to be set so to increase the legibility of the codebase and aid in validating the `require` check's condition.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The referenced `require` error remains without an explicit error message or in-line documentation justifying it, rendering this exhibit acknowledged.

ASS-04C: Loop Iterator Optimization

Type	Severity	Location
Gas Optimization	● Informational	AsSequentialSet.sol:L250

Description:

The linked `for` loop increments / decrements the iterator "safely" due to Solidity's built-in safe arithmetics (post-`0.8.x`).

Example:

```
src/libs/AsSequentialSet.sol
```

```
SOL
```

```
250 for (uint256 i = 0; i < q.data.length; i++) {
```

Recommendation:

We advise the increment / decrement operation to be performed in an `unchecked` code block as the last statement within the `for` loop to optimize its execution cost.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The referenced loop iterator's increment statement has been relocated at the end of the `for` loop's body and has been unwrapped in an `unchecked` code block, optimizing its gas cost.

ASS-05C: Redundant Deletion Operation

Type	Severity	Location
Gas Optimization	Informational	AsSequentialSet.sol:L130

Description:

The referenced `delete` operation is redundant as the data entry is overwritten in the ensuing statement.

Example:

src/libs/AsSequentialSet.sol

```
SOL

127 function removeAt(Set storage q, uint256 i) internal {
128     require(i < q.data.length, "Index out of bounds");
129     if (i < q.data.length - 1) {
130         delete q.data[i];
131         q.data[i] = q.data[q.data.length - 1];
132     }
133     q.data.pop();
134 }
```

Recommendation:

We advise the `delete` operation to be omitted, optimizing the code's gas cost.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The redundant `delete` operation has been safely omitted from the codebase, optimizing the function's gas cost.

ChainlinkUtils Code Style Findings

CUS-01C: Ineffectual Usage of Safe Arithmetics

Type	Severity	Location
Language Specific	Informational	ChainlinkUtils.sol:L29, L30

Description:

The linked mathematical operation is guaranteed to be performed safely by surrounding conditionals evaluated in either `require` checks or `if-else` constructs.

Example:

```
src/libs/ChainlinkUtils.sol
```

```
SOL
```

```
27 // debase pyth feed decimals to target decimals
28 return _targetDecimals >= feedDecimals ?
29     uint256(basePrice) * 10 ** uint32(_targetDecimals - feedDecimals) :
30     uint256(basePrice) / 10 ** uint32(feedDecimals - _targetDecimals);
```

Recommendation:

Given that safe arithmetics are toggled on by default in `pragma` versions of `0.8.x`, we advise the linked statement to be wrapped in an `unchecked` code block thereby optimizing its execution cost.

Alleviation (59b75fbee1):

The relevant statement has been significantly refactored and now lives under the

`ChainlinkProvider::toUsdBp`, wrapped in an `unchecked` code block.

We do not consider the present `unchecked` code block introduced to be safe, as it relies on an `_invert` flag instead of the actual relation between the variables subtracted thus rendering this exhibit **not validated** to highlight the fact of this insecurity.

Alleviation (efbeab6478):

The Astrolab DAO team opted to revert the `unchecked` code block's introduction, ensuring that the statements are performed safely yet inefficiently per their original implementation.

As such, we consider this exhibit acknowledged as the Astrolab DAO team does not intend to apply the optimization properly.

CUS-02C: Repetitive Value Literal

Type	Severity	Location
Code Style	● Informational	ChainlinkUtils.sol:L49, L50

Description:

The linked value literal is repeated across the codebase multiple times.

Example:

```
src/libs/ChainlinkUtils.sol
```

```
SOL
```

```
49  return getPriceUsd(_feeds[0], _validities[0], 18)
```

Recommendation:

We advise it to be set to a `constant` variable instead optimizing the legibility of the codebase.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The referenced value literal now lives under the `PriceProvider` implementation and specifically the `USD_DECIMALS` constant variable, addressing this exhibit.

PythUtils Code Style Findings

PUS-01C: Ineffectual Usage of Safe Arithmetics

Type	Severity	Location
Language Specific	● Informational	PythUtils.sol:L33, L34

Description:

The linked mathematical operation is guaranteed to be performed safely by surrounding conditionals evaluated in either `require` checks or `if-else` constructs.

Example:

```
src/libs/PythUtils.sol
```

```
SOL

31 // debase pyth feed decimals to target decimals
32 return _targetDecimals >= feedDecimals ?
33     basePrice * 10 ** uint32(_targetDecimals - feedDecimals) :
34     basePrice / 10 ** uint32(feedDecimals - _targetDecimals);
```

Recommendation:

Given that safe arithmetics are toggled on by default in `pragma` versions of `0.8.x`, we advise the linked statement to be wrapped in an `unchecked` code block thereby optimizing its execution cost.

Alleviation (59b75fbee1):

The relevant statement has been significantly refactored and now lives under the

`PythProvider::_toUsdBp`, wrapped in an `unchecked` code block.

We do not consider the present `unchecked` code block introduced to be safe, as it relies on an `_invert` flag instead of the actual relation between the variables subtracted thus rendering this exhibit **not validated** to highlight the fact of this insecurity.

Alleviation (efbeab6478):

The Astrolab DAO team opted to revert the `unchecked` code block's introduction, ensuring that the statements are performed safely yet inefficiently per their original implementation.

As such, we consider this exhibit acknowledged as the Astrolab DAO team does not intend to apply the optimization properly.

PUS-02C: Repetitive Value Literal

Type	Severity	Location
Code Style	● Informational	PythUtils.sol:L70, L71

Description:

The linked value literal is repeated across the codebase multiple times.

Example:

```
src/libs/PythUtils.sol
SOL
70  return getPriceUsd(_pyth, _feeds[0], _validities[0], 18)
```

Recommendation:

We advise it to be set to a `constant` variable instead optimizing the legibility of the codebase.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The referenced value literal now lives under the `PriceProvider` implementation and specifically the `USD_DECIMALS` constant variable, addressing this exhibit.

StrategyV5 Code Style Findings

SV5-01C: Generic Typographic Mistake

Type	Severity	Location
Code Style	Informational	StrategyV5.sol:L467, L474

Description:

The referenced line contains a typographical mistake (i.e. `private` variable without an underscore prefix) or generic documentational error (i.e. copy-paste) that should be corrected.

Example:

```
src/abstract/StrategyV5.sol
```

```
SOL
```

```
467 * @param _amount Amount of asset to liquidate with (0 == totalPendingAssetRequest()  
+ allocated.bp(100))
```

Recommendation:

We advise this to be done so to enhance the legibility of the codebase.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The documentation was updated to reflect the `150` number that used to be utilized in the instance of the codebase during the preliminary report, however, the latest instance utilizes the `50` value as an overallocation.

As such, we advise the documentation to be updated so as to reflect this adjustment.

SV5-02C: Improper Declarations of Abstract Functions

Type	Severity	Location
Standard Conformity	● Informational	StrategyV5.sol:L109-L112, L160-L162, L238-L241, L309, L316

Description:

The referenced functions are meant to be `virtual` and implemented by derivative implementations, however, an empty declaration is present in both that would permit each to be invoked.

Example:

src/abstract/StrategyV5.sol

SOL

```
103 /**
104  * @notice Withdraw asset function, can remove all funds in case of emergency
105  * @param _amounts Amounts of asset to withdraw
106  * @param _params Swaps calldata
107  * @return assetsRecovered Amount of asset withdrawn
108 */
109 function _liquidate(
110     uint256[8] calldata _amounts, // from previewLiquidate()
111     bytes[] memory _params
112 ) internal virtual returns (uint256 assetsRecovered) {}
```

Recommendation:

We advise the functions to be declared without a code block (`{ }`) to ensure they are overridden by derivative implementations.

Alleviation (59b75fbee1):

While some functions have been properly implemented thus rendering the empty code block observation no longer applicable, functions such as `StrategyV5::stake` remain with an empty code block rendering this exhibit partially alleviated.

Alleviation (efbeab6478):

All functions have been properly updated to no longer implement a code block where applicable, rendering this exhibit fully addressed.

SV5-03C: Ineffectual Usage of Safe Arithmetics

Type	Severity	Location
Language Specific	Informational	StrategyV5.sol:L484, L507

Description:

The linked mathematical operation is guaranteed to be performed safely by surrounding conditionals evaluated in either `require` checks or `if-else` constructs.

Example:

```
src/abstract/StrategyV5.sol
```

```
SOL
```

```
481 if (need > _amount)
482     need = _amount;
483 amounts[i] = _assetToInput(need, i);
484 _amount -= need;
```

Recommendation:

Given that safe arithmetics are toggled on by default in `pragma` versions of `0.8.x`, we advise the linked statement to be wrapped in an `unchecked` code block thereby optimizing its execution cost.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

An `unchecked` code block has been safely introduced in both referenced instances, optimizing the code's gas cost.

SV5-04C: Inefficient Iterator Type

Type	Severity	Location
Gas Optimization	Informational	StrategyV5.sol:L186, L381, L432, L461, L477, L500

Description:

The referenced `for` loops utilize a `uint8` variable as an iterator which is inefficient.

Example:

```
src/abstract/StrategyV5.sol
```

```
SOL
```

```
186 for (uint8 i = 0; i < rewardLength; i++) {
```

Recommendation:

As the EVM is built to operate on 32-byte (256-bit) data types, we advise the iterator types to be bumped to `uint256`, optimizing their gas cost.

Alleviation (59b75fbee1):

Most of the instances have been properly upcast to optimize them, however, the `strategyV5::_invest` and `StrategyV5::_liquidate` functions continue to use suboptimal operators rendering this exhibit partially alleviated.

Alleviation (efbeab6478):

The `uint8` iterators in the `strategyV5::_invest` and `StrategyV5::_liquidate` functions have been updated to their upcasted format, applying the described optimization in full.

SV5-05C: Loop Iterator Optimizations

Type	Severity	Location
Gas Optimization	Informational	StrategyV5.sol:L186, L381, L432, L461, L477, L500

Description:

The linked `for` loops increment / decrement their iterator "safely" due to Solidity's built - in safe arithmetics (post-0.8.x).

Example:

```
src/abstract/StrategyV5.sol
```

```
SOL
```

```
186 for (uint8 i = 0; i < rewardLength; i++) {
```

Recommendation:

We advise the increment / decrement operations to be performed in an `unchecked` code block as the last statement within each `for` loop to optimize their execution cost.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The referenced loop iterator increment statements have been relocated at the end of each respective `for` loop's body and have been unwrapped in an `unchecked` code block, optimizing their gas cost.

SV5-06C: Redundant Application of Access Control

Type	Severity	Location
Gas Optimization	Informational	StrategyV5.sol:L298

Description:

The `StrategyV5::compound` function redundantly applies the `AsManageable::onlyKeeper` modifier when its inner calls will perform the same validation (`StrategyV5::compound -> StrategyV5::_compound -> StrategyV5::harvest -> StrategyV5::_harvest -> StrategyV5::_swapRewards`).

Example:

src/abstract/StrategyV5.sol

```
SOL

286 /**
287  * @notice Executes the compound operation in the strategy
288  * @param _amounts Amounts of inputs to compound (in asset, after harvest->
should include rewards)
289  * @param _params Generic callData for the compound operation
290  * @return iouReceived IOUs received from the compound operation
291  * @return harvestedRewards Amount of rewards harvested
292 */
293 function compound(
294     uint256[8] calldata _amounts,
295     bytes[] memory _params
```

Example (Cont.):

```
SOL
296 )
297     external
298     onlyKeeper
299     returns (uint256 iouReceived, uint256 harvestedRewards)
300 {
301     (iouReceived, harvestedRewards) = _compound(_amounts, _params);
302     emit Compound(iouReceived, block.timestamp);
303 }
```

Recommendation:

We advise access control to be solely applied to the innermost functions, ensuring that restrictions are optimally applied.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The access control inefficiency remains in the codebase rendering this exhibit acknowledged.

SV5-07C: Redundant Parenthesis Statement

Type	Severity	Location
Code Style	Informational	StrategyV5.sol:L147

Description:

The referenced statement is redundantly wrapped in parenthesis `(())`.

Example:

```
src/abstract/StrategyV5.sol
```

```
SOL
```

```
147 if ((liquidityAvailable < _minLiquidity) && !_panic)
```

Recommendation:

We advise them to be safely omitted, increasing the legibility of the codebase.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The redundant parenthesis in the referenced statement have been safely omitted.

StrategyV5Abstract Code Style Findings

SVT-01C: Generic Typographic Mistakes

Type	Severity	Location
Code Style	Informational	StrategyV5Abstract.sol:L38, L41, L44, L45, L46, L52

Description:

The referenced lines contain typographical mistakes (i.e. `private` variable without an underscore prefix) or generic documentational errors (i.e. copy-paste) that should be corrected.

Example:

```
src/abstract/StrategyV5Abstract.sol
SOL
38 address internal stratProxy; // Address of the strategy proxy
```

Recommendation:

We advise them to be corrected enhancing the legibility of the codebase.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The referenced variable as well as its associated concept have been removed from the codebase rendering this exhibit no longer applicable.

StrategyV5Agent Code Style Findings

SVA-01C: Inefficient Iterator Type

Type	Severity	Location
Gas Optimization	Informational	StrategyV5Agent.sol:L125, L142

Description:

The referenced `for` loops utilize a `uint8` variable as an iterator which is inefficient.

Example:

```
src/abstract/StrategyV5Agent.sol
```

```
SOL
```

```
142 for (uint8 i = 0; i < _rewardTokens.length; i++) {
```

Recommendation:

As the EVM is built to operate on 32-byte (256-bit) data types, we advise the iterator types to be bumped to `uint256`, optimizing their gas cost.

Alleviation (59b75fbee1):

While the referenced iterator types have been optimized, the `StrategyV5Agent::_setInputWeights` function continues to utilize a `uint8` iterator type which is inefficient.

Alleviation (efbeab6478):

The `uint8` iterator in the `StrategyV5Agent::_setInputWeights` function has been updated accordingly, rendering this exhibit fully addressed.

SVA-02C: Loop Iterator Optimizations

Type	Severity	Location
Gas Optimization	Informational	StrategyV5Agent.sol:L51, L55, L125, L142

Description:

The linked `for` loops increment / decrement their iterator "safely" due to Solidity's built - in safe arithmetics (post-0.8.x).

Example:

```
src/abstract/StrategyV5Agent.sol
```

```
SOL
```

```
51  for (uint256 i = 0; i < rewardLength; i++) {
```

Recommendation:

We advise the increment / decrement operations to be performed in an `unchecked` code block as the last statement within each `for` loop to optimize their execution cost.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The referenced loop iterator increment statements have been relocated at the end of each respective `for` loop's body and have been unwrapped in an `unchecked` code block, optimizing their gas cost.

StrategyV5Chainlink Code Style Findings

SVC-01C: Generic Typographic Mistake

Type	Severity	Location
Code Style	● Informational	StrategyV5Chainlink.sol:L24

Description:

The referenced line contains a typographical mistake (i.e. `private` variable without an underscore prefix) or generic documentational error (i.e. copy-paste) that should be corrected.

Example:

```
src/abstract/StrategyV5Chainlink.sol
SOL
24 mapping (IChainlinkAggregatorV3 => uint8) internal decimalsByFeed;
```

Recommendation:

We advise this to be done so to enhance the legibility of the codebase.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The relevant declaration, now located under `PriceProvider`, has been properly prefixed with an underscore rendering this exhibit addressed.

SVC-02C: Loop Iterator Optimization

Type	Severity	Location
Gas Optimization	Informational	StrategyV5Chainlink.sol:L69

Description:

The linked `for` loop increments / decrements the iterator "safely" due to Solidity's built-in safe arithmetics (post-`0.8.x`).

Example:

src/abstract/StrategyV5Chainlink.sol

SOL

```
69  for (uint256 i = 0; i < _chainlinkParams.inputFeeds.length; i++) {
```

Recommendation:

We advise the increment / decrement operation to be performed in an `unchecked` code block as the last statement within the `for` loop to optimize its execution cost.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The relevant loop has been relocated under the `PriceProvider::_setFeeds` function and its iterator has been optimized rendering this exhibit addressed.

SVC-03C: Repetitive Value Literal

Type	Severity	Location
Code Style	Informational	StrategyV5Chainlink.sol:L91, L93

Description:

The linked value literal is repeated across the codebase multiple times.

Example:

```
src/abstract/StrategyV5Chainlink.sol
SOL
91 uint256 retiredPrice = ChainlinkUtils.getPriceUsd(retiredFeed,
validityByFeed[retiredFeed], 18);
```

Recommendation:

We advise it to be set to a `constant` variable instead optimizing the legibility of the codebase.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The relevant literal has been declared as a `constant` labelled `USD_DECIMALS` under the `PriceProvider` implementation, addressing this exhibit.

StrategyV5Pyth Code Style Findings

SVP-01C: Generic Typographic Mistake

Type	Severity	Location
Code Style	● Informational	StrategyV5Pyth.sol:L26

Description:

The referenced line contains a typographical mistake (i.e. `private` variable without an underscore prefix) or generic documentational error (i.e. copy-paste) that should be corrected.

Example:

```
src/abstract/StrategyV5Pyth.sol
SOL
26 IPythAggregator internal pyth; // Pyth oracle
```

Recommendation:

We advise this to be done so to enhance the legibility of the codebase.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The relevant declaration, now located under `PythProvider`, has been properly prefixed with an underscore rendering this exhibit addressed.

SVP-02C: Loop Iterator Optimizations

Type	Severity	Location
Gas Optimization	Informational	StrategyV5Pyth.sol:L73, L126

Description:

The linked `for` loops increment / decrement their iterator "safely" due to Solidity's built - in safe arithmetics (post-`0.8.x`).

Example:

```
src/abstract/StrategyV5Pyth.sol
```

```
SOL
```

```
73  for (uint256 i = 0; i < _pythParams.inputFeeds.length; i++) {
```

Recommendation:

We advise the increment / decrement operations to be performed in an `unchecked` code block as the last statement within each `for` loop to optimize their execution cost.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The former loop has been relocated under the `PriceProvider::_setFeeds` function and its iterator has been optimized whilst the latter loop is no longer present in the codebase.

These actions cumulatively render this exhibit addressed.

SVP-03C: Repetitive Value Literal

Type	Severity	Location
Code Style	● Informational	StrategyV5Pyth.sol:L99, L107

Description:

The linked value literal is repeated across the codebase multiple times.

Example:

```
src/abstract/StrategyV5Pyth.sol
SOL
99 18);
```

Recommendation:

We advise it to be set to a `constant` variable instead optimizing the legibility of the codebase.

Alleviation (59b75fbe1d8f3dee807c928f18be41c58b904e1):

The relevant literal has been declared as a `constant` labelled `USD_DECIMALS` under the `PriceProvider` implementation, addressing this exhibit.

Finding Types

A description of each finding type included in the report can be found below and is linked by each respective finding. A full list of finding types Omnisca has defined will be viewable at the central audit methodology we will publish soon.

Input Sanitization

As there are no inherent guarantees to the inputs a function accepts, a set of guards should always be in place to sanitize the values passed in to a particular function.

Indeterminate Code

These types of issues arise when a linked code segment may not behave as expected, either due to mistyped code, convoluted if blocks, overlapping functions / variable names and other ambiguous statements.

Language Specific

Language specific issues arise from certain peculiarities that the Circom language boasts that discerns it from other conventional programming languages.

Curve Specific

Circom defaults to using the BN128 scalar field (a 254-bit prime field), but it also supports BSL12-381 (which has a 255-bit scalar field) and Goldilocks (with a 64-bit scalar field). However, since there are no constants denoting either the prime or the prime size in bits available in the Circom language, some Circomlib templates like `sign` (which returns the sign of the input signal), and `AliasCheck` (used by the strict versions of `Num2Bits` and `Bits2Num`), hardcode either the BN128 prime size or some other constant related to BN128. Using these circuits with a custom prime may thus lead to unexpected results and should be avoided.

Code Style

In these types of findings, we identify whether a project conforms to a particular naming convention and whether that convention is consistent within the codebase and legible. In case of inconsistencies, we point them out under this category. Additionally, variable shadowing falls under this category as well which is identified when a local-level variable contains the same name as a toplevel variable in the circuit.

Mathematical Operations

This category is used when a mathematical issue is identified. This implies an issue with the implementation of a calculation compared to the specifications.

Logical Fault

This category is a bit broad and is meant to cover implementations that contain flaws in the way they are implemented, either due to unimplemented functionality, unaccounted-for edge cases or similar extraordinary scenarios.

Privacy Concern

This category is used when information that is meant to be kept private is made public in some way.

Proof Concern

Under-constrained signals are one of the most common issues in zero-knowledge circuits. Issues with proof generation fall under this category.

Severity Definition

In the ever-evolving world of blockchain technology, vulnerabilities continue to take on new forms and arise as more innovative projects manifest, new blockchain-level features are introduced, and novel layer-2 solutions are launched. When performing security reviews, we are tasked with classifying the various types of vulnerabilities we identify into subcategories to better aid our readers in understanding their impact.

Within this page, we will clarify what each severity level stands for and our approach in categorizing the findings we pinpoint in our audits. To note, all severity assessments are performed **as if the contract's logic cannot be upgraded** regardless of the underlying implementation.

Severity Levels

There are five distinct severity levels within our reports; `unknown`, `informational`, `minor`, `medium`, and `major`. A TL;DR overview table can be found below as well as a dedicated chapter to each severity level:

	Impact (None)	Impact (Low)	Impact (Moderate)	Impact (High)
Likelihood (None)	Informational	Informational	Informational	Informational
Likelihood (Low)	Informational	Minor	Minor	Medium
Likelihood (Moderate)	Informational	Minor	Medium	Major
Likelihood (High)	Informational	Medium	Major	Major

Unknown Severity

The `unknown` severity level is reserved for misbehaviors we observe in the codebase that cannot be quantified using the above metrics. Examples of such vulnerabilities include potentially desirable system behavior that is undocumented, reliance on external dependencies that are out-of-scope but could result in some form of vulnerability arising, use of external out-of-scope contracts that appears incorrect but cannot be pinpointed, and other such vulnerabilities.

In general, `unknown` severity level vulnerabilities require follow-up information by the project being audited and are either adjusted in severity (if valid), or marked as nullified (if invalid).

Additionally, the `unknown` severity level is sometimes assigned to centralization issues that cannot be assessed in likelihood due to their exploitation being tied to the honesty of the project's team.

Informational Severity

The `informational` severity level is dedicated to findings that do not affect the code functionally and tend to be stylistic or optimizational in nature. Certain edge cases are also set under `informational` vulnerabilities, such as overflow operations that will not manifest in the lifetime of the contract but should be guarded against as a best practice, to give an example.

Minor Severity

The `minor` severity level is meant for vulnerabilities that require functional changes in the code but tend to either have little impact or be unlikely to be recreated in a production environment. These findings can be acknowledged except for findings with a moderate impact but low likelihood which must be alleviated.

Medium Severity

The `medium` severity level is assigned to vulnerabilities that must be alleviated and have an observable impact on the overall project. These findings can only be acknowledged if the project deems them desirable behavior and we disagree with their point-of-view, instead urging them to reconsider their stance while marking the exhibit as acknowledged given that the project has ultimate say as to what vulnerabilities they end up patching in their system.

Major Severity

The `major` severity level is the maximum that can be specified for a finding and indicates a significant flaw in the code that must be alleviated.

Likelihood & Impact Assessment

As the preface chapter specifies, the blockchain space is constantly reinventing itself meaning that new vulnerabilities take place and our understanding of what security means differs year-to-year.

In order to reliably assess the likelihood and impact of a particular vulnerability, we instead apply an abstract measurement of a vulnerability's impact, duration the impact is applied for, and probability that the vulnerability would be exploited in a production environment.

Our proposed definitions are inspired by multiple sources in the security community and are as follows:

Disclaimer

The following disclaimer applies to all versions of the audit report produced (preliminary / public / private) and is in effect for all past, current, and future audit reports that are produced and hosted under Omniscia:

IMPORTANT TERMS & CONDITIONS REGARDING OUR SECURITY AUDITS/REVIEWS/REPORTS AND ALL PUBLIC/PRIVATE CONTENT/DELIVERABLES

Omniscia ("Omniscia") has conducted an independent security review to verify the integrity of and highlight any vulnerabilities, bugs or errors, intentional or unintentional, that may be present in the codebase that were provided for the scope of this Engagement.

Blockchain technology and the cryptographic assets it supports are nascent technologies. This makes them extremely volatile assets. Any assessment report obtained on such volatile and nascent assets may include unpredictable results which may lead to positive or negative outcomes.

In some cases, services provided may be reliant on a variety of third parties. This security review does not constitute endorsement, agreement or acceptance for the Project and technology that was reviewed. Users relying on this security review should not consider this as having any merit for financial advice or technological due diligence in any shape, form or nature.

The veracity and accuracy of the findings presented in this report relate solely to the proficiency, competence, aptitude and discretion of our auditors. Omniscia and its employees make no guarantees, nor assurance that the contracts are free of exploits, bugs, vulnerabilities, depreciation of technologies or any system / economical / mathematical malfunction.

This audit report shall not be printed, saved, disclosed nor transmitted to any persons or parties on any objective, goal or justification without due written assent, acquiescence or approval by Omniscia.

All the information/opinions/suggestions provided in this report does not constitute financial or investment advice, nor should it be used to signal that any person reading this report should invest their funds without sufficient individual due diligence regardless of the findings presented in this report.

Information in this report is provided 'as is'. Omniscia is under no covenant to the completeness, accuracy or solidity of the contracts reviewed. Omniscia's goal is to help reduce the attack vectors/surface and the high level of variance associated with utilizing new and consistently changing technologies.

Omniscia in no way claims any guarantee, warranty or assurance of security or functionality of the technology that was in scope for this security review.

In no event will Omniscia, its partners, employees, agents or any parties related to the design/creation of this security review be ever liable to any parties for, or lack thereof, decisions and/or actions with regards to the information provided in this security review.

Cryptocurrencies and all other technologies directly or indirectly related to cryptocurrencies are not standardized, highly prone to malfunction and extremely speculative by nature. No due diligence and/or safeguards may be insufficient and users should exercise maximum caution when participating and/or investing in this nascent industry.

The preparation of this security review has made all reasonable attempts to provide clear and actionable recommendations to the Project team (the "client") with respect to the rectification, amendment and/or revision of any highlighted issues, vulnerabilities or exploits within the contracts in scope for this engagement.

It is the sole responsibility of the Project team to provide adequate levels of test and perform the necessary checks to ensure that the contracts are functioning as intended, and more specifically to ensure that the functions contained within the contracts in scope have the desired intended effects, functionalities and outcomes, as documented by the Project team.

All services, the security reports, discussions, work product, attack vectors description or any other materials, products or results of this security review engagement is provided "as is" and "as available" and with all faults, uncertainty and defects without warranty or guarantee of any kind.

Omniscia will assume no liability or responsibility for delays, errors, mistakes, or any inaccuracies of content, suggestions, materials or for any loss, delay, damage of any kind which arose as a result of this engagement/security review.

Omniscia will assume no liability or responsibility for any personal injury, property damage, of any kind whatsoever that resulted in this engagement and the customer having access to or use of the products, engineers, services, security report, or any other other materials.

For avoidance of doubt, this report, its content, access, and/or usage thereof, including any associated services or materials, shall not be considered or relied upon as any form of financial, investment, tax, legal, regulatory, or any other type of advice.