



# Cyberscope

## Audit Report

# BeraTrax

March 2025

Files      Vault, ArberaZapper, InfraredZapper, KodiakZapper, SteerZapper, ZapperBase, LpRouter, SwapRouter, ArberaStrategy, InfraredStrategy, KodiakStrategy, SteerStrategy, SInfraredFactory, StrategyBase, ArberaFactory, KodiakFactory, SteerFactory, StrategyFactoryBase, VaultFactory, ControllerFactory, Controller

Audited by   © cyberscope

# Table of Contents

<b>Table of Contents</b>	<b>1</b>
<b>Risk Classification</b>	<b>3</b>
<b>Review</b>	<b>4</b>
Audit Updates	4
Source Files	4
<b>Overview</b>	<b>6</b>
Controller	6
Factories	6
Strategies	6
Vault	7
Zappers	7
Disclaimer	7
<b>Findings Breakdown</b>	<b>8</b>
<b>Diagnostics</b>	<b>9</b>
PDI - Potential Denominator Inconsistency	11
Description	11
Recommendation	11
ILU - Inconsistent Library Use	12
Description	12
Recommendation	13
PRE - Potential Reentrance Exploit	14
Description	14
Recommendation	15
CCR - Contract Centralization Risk	16
Description	16
Recommendation	19
ELFM - Exceeds Fees Limit	20
Description	20
Recommendation	21
GRLNS - Gamma Remove Liquidity Not Supported	22
Description	22
Recommendation	22
MC - Missing Check	23
Description	23
Recommendation	23
MN - Misspelled Naming	24
Description	24
Recommendation	25
ORA - Overwriting Rewards Amount	26

Description	26
Recommendation	26
PLPI - Potential Liquidity Provision Inadequacy	27
Description	27
Recommendation	29
SAU - Swapped Amount Uninitialized	30
Description	30
Recommendation	30
UBUFL - Unintended Balance Used For Liquidity	31
Description	31
Recommendation	32
UTPD - Unverified Third Party Dependencies	33
Description	33
Recommendation	34
L04 - Conformance to Solidity Naming Conventions	35
Description	35
Recommendation	35
L09 - Dead Code Elimination	36
Description	36
Recommendation	37
L14 - Uninitialized Variables in Local Scope	38
Description	38
Recommendation	38
L15 - Local Scope Variable Shadowing	39
Description	39
Recommendation	39
L17 - Usage of Solidity Assembly	40
Description	40
Recommendation	40
<b>Functions Analysis</b>	<b>41</b>
<b>Inheritance Graph</b>	<b>59</b>
<b>Summary</b>	<b>60</b>
<b>Disclaimer</b>	<b>61</b>
<b>About Cyberscope</b>	<b>62</b>

## Risk Classification

The criticality of findings in Cyberscope's smart contract audits is determined by evaluating multiple variables. The two primary variables are:

1. **Likelihood of Exploitation:** This considers how easily an attack can be executed, including the economic feasibility for an attacker.
2. **Impact of Exploitation:** This assesses the potential consequences of an attack, particularly in terms of the loss of funds or disruption to the contract's functionality.

Based on these variables, findings are categorized into the following severity levels:

1. **Critical:** Indicates a vulnerability that is both highly likely to be exploited and can result in significant fund loss or severe disruption. Immediate action is required to address these issues.
2. **Medium:** Refers to vulnerabilities that are either less likely to be exploited or would have a moderate impact if exploited. These issues should be addressed in due course to ensure overall contract security.
3. **Minor:** Involves vulnerabilities that are unlikely to be exploited and would have a minor impact. These findings should still be considered for resolution to maintain best practices in security.
4. **Informative:** Points out potential improvements or informational notes that do not pose an immediate risk. Addressing these can enhance the overall quality and robustness of the contract.

Severity	Likelihood / Impact of Exploitation
● Critical	Highly Likely / High Impact
● Medium	Less Likely / High Impact or Highly Likely/ Lower Impact
● Minor / Informative	Unlikely / Low to no Impact

# Review

## Audit Updates

Initial Audit	17 Jan 2025 <a href="https://github.com/cyberscope-io/audits/blob/main/c5-btx/v1/audit.pdf">https://github.com/cyberscope-io/audits/blob/main/c5-btx/v1/audit.pdf</a>
Corrected Phase 2	31 Jan 2025 <a href="https://github.com/cyberscope-io/audits/blob/main/c5-btx/v2/audit.pdf">https://github.com/cyberscope-io/audits/blob/main/c5-btx/v2/audit.pdf</a>
Corrected Phase 3	09 Mar 2025 <a href="https://github.com/cyberscope-io/audits/blob/main/c5-btx/v3/audit.pdf">https://github.com/cyberscope-io/audits/blob/main/c5-btx/v3/audit.pdf</a>
Corrected Phase 4	28 Mar 2025

## Source Files

Filename	SHA256
controllers/Controller.sol	d1dbd00e30ebfc4ce21b48d43c5799d6593c2b70a878358068e759a9d468616b
factories/ArberaFactory.sol	37320f3de510b1e8726a1fcb22f23145e33779ee48618dd8df1997bc3532bfff
factories/ControllerFactory.sol	eed63c88d2896f9f85db51e50df0024b3890fd74fb834d141c03852321d48cfc
factories/InfraredFactory.sol	86b2418582f27a17e22cf117b3b252becc9c8464a57b2ba20cc816941faafaac
factories/KodiakFactory.sol	a745d58121f1d2ec887ba7460f33ac078ae1713cf11794c6b9ebb761a5aa2531
factories/SteerFactory.sol	6aac96481f34974bf86849962d3c10fa201ac66b7f7ff9effb571ee5bd9e76ca
factories/StrategyFactoryBase.sol	4c293174ad20551ba5d31c1273c818d61d725c135ffab9e3e26641e786d6329c

<b>factories/VaultFactory.sol</b>	e9230c9b34c80543b9eaa2e6906d90fa43bfa12346775f8b1c8c1a5fd0ff32c4
<b>strategies/ArberaStrategy.sol</b>	d780c7c63a8ea516fe926bc765fe321e9db4dc6a01c77d4875769178b965ef56
<b>strategies/InfraredStrategy.sol</b>	3de78e266f673d385fca220f6cd6d9f1c28407a80ea8605d009b46370fe89402
<b>strategies/KodiakStrategy.sol</b>	289809d4c4937f7d0573528ad5f38ae89b3e440680c95ca69985e61ce938b234
<b>strategies/SteerStrategy.sol</b>	3ad54d64918c52ca51d99927de12d2aea0fa7bf7d78758fc35117622e6a31c99
<b>strategies/StrategyBase.sol</b>	9b60c9f9823592ebf21dfec4cc4cc35cbde3acd28189d68d34d41c607bfb653a
<b>utils/LpRouter.sol</b>	294fa55da1f68953be23088242d3ba6769462f0b70a2b06df8c6215fd57fe818
<b>utils/SwapRouter.sol</b>	eef05d4e8128803e0ed4730381f3ef3a8e40785d3cac4469e100998569069716
<b>vaults/Vault.sol</b>	381b3de538cf888fe733cfbfc6f370a618ab3b3a12f2253f9ebb23b1dab925f1
<b>zappers/ArberaZapper.sol</b>	a210ac82f8ad9120124bca01764b0b9cd40ddd2dbe7ccd98799daf3826138bd4
<b>zappers/InfraredZapper.sol</b>	0a95a08e2c9af9eb084c89e928fa72b34730d32c1337e88e39313af47a679db9
<b>zappers/KodiakZapper.sol</b>	fb161695f9d1e8b7519ed37facb56dd90bb8192a3a2251efce982328dae24630
<b>zappers/SteerZapper.sol</b>	c29d78619ee6854df959cb8d3d031840913d18dd4e4b1300722b4e0a669c02f6
<b>zappers/ZapperBase.sol</b>	0615f8a0a44743aeac685fe41ef113c753f9e0da694def904c37bdbbae9e4f94

## Overview

The contracts implement a modular and efficient yield aggregator built on the Ethereum Virtual Machine (EVM). Contrax optimises yield generation by utilising a combination of smart contracts that automate asset management and maximise returns. The dApp provides users with a seamless platform to deposit their assets and earn competitive APR. Its architecture is designed for scalability, security, and flexibility, ensuring that users can confidently participate in an efficient and transparent ecosystem for yield farming.

## Controller

The Controller serves as the central management layer connecting the Vaults to their respective investment Strategies. Its primary purpose is to oversee the flow of funds from the Vaults to the Strategies and ensure that assets are optimally allocated to generate yield. The Controller maintains control over approved Strategies, allowing governance to update or revoke them as needed to adapt to changing market conditions. By acting as an intermediary, the Controller provides a secure and modular architecture, separating user deposits from the underlying investment logic while enforcing robust access controls and operational flexibility.

## Factories

The Factory is responsible for the deployment and initialization of Vaults and Controllers, enabling a seamless creation process for these core components. By standardizing the deployment of Vault-Controller pairs, the Factory ensures consistency in configuration and governance integration. It allows for scalability and adaptability by enabling developers and strategists to deploy new Vaults and Controllers for different assets, while ensuring the system adheres to predefined rules and relationships. As a central hub for new deployments, the Factory streamlines the expansion of the protocol while maintaining security and governance integrity.

## Strategies

The Strategies are the yield-generating engines of the system, implementing specific logic for investing assets into staking pools, liquidity farms, or other financial products. Their goal is to maximize returns on assets deposited by the Vaults while adhering to risk and

operational constraints defined by governance. Each Strategy is tailored to a specific investment opportunity and can be updated or replaced as market conditions evolve. By abstracting investment logic, Strategies provide flexibility and modularity, allowing the protocol to adapt quickly to new yield sources without impacting the user-facing components of the system.

## Vault

The Vault is the user-facing contract that manages deposits, withdrawals, and the representation of user stakes through Vault shares. When users deposit assets into the Vault, they receive shares that represent their proportional claim to the total assets under management. The Vault works closely with the Controller to allocate idle assets to Strategies through the `earn` function, ensuring that deposits are continuously put to productive use. During withdrawals, the Vault redeems shares for assets, either using its own reserves or retrieving funds from the Controller. By acting as a secure intermediary, the Vault simplifies user interactions while managing the complexities of yield generation.

## Zappers

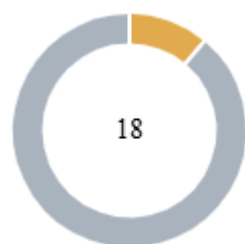
The Zappers are designed to enhance user convenience by automating asset conversions and liquidity provision for deposits and withdrawals. They allow users to interact with Vaults using various tokens, handling the necessary swaps, liquidity additions, and token approvals behind the scenes. Zappers streamline the process of entering and exiting Vaults, eliminating the need for users to manually manage token conversions or intermediate steps. By abstracting away these complexities, Zappers improve accessibility and usability, enabling a broader range of users to participate in the protocol's yield-generating ecosystem.

## Disclaimer

@spherex-xyz/contracts, being an external source, are considered out of scope for this review and will not be analyzed as part of this assessment.



## Findings Breakdown



Critical	0
Medium	2
Minor / Informative	16

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	2	0	0	0
Minor / Informative	16	0	0	0

## Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	PDI	Potential Denominator Inconsistency	Unresolved
●	ILU	Inconsistent Library Use	Unresolved
●	PRE	Potential Reentrance Exploit	Unresolved
●	CCR	Contract Centralization Risk	Unresolved
●	ELFM	Exceeds Fees Limit	Unresolved
●	GRLNS	Gamma Remove Liquidity Not Supported	Unresolved
●	MC	Missing Check	Unresolved
●	MN	Misspelled Naming	Unresolved
●	ORA	Overwriting Rewards Amount	Unresolved
●	PLPI	Potential Liquidity Provision Inadequacy	Unresolved
●	SAU	Swapped Amount Uninitialized	Unresolved
●	UBUFL	Unintended Balance Used For Liquidity	Unresolved
●	UTPD	Unverified Third Party Dependencies	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved

●	L09	Dead Code Elimination	Unresolved
●	L14	Uninitialized Variables in Local Scope	Unresolved
●	L15	Local Scope Variable Shadowing	Unresolved
●	L17	Usage of Solidity Assembly	Unresolved

## PDI - Potential Denominator Inconsistency

Criticality	Medium
Location	LpRouter.sol#L30
Status	Unresolved

### Description

`MAX_DENOMINATOR` is a constant variable equal to `100_000`. It is used to calculate the `amountIn` by subtracting the fees in `_getQuoteV3` and `_getQuoteV3WithPath`. However if the denominator used in the pool is different the calculation will be wrong.

```
uint256 public constant MAX_DENOMINATOR = 100000;

function _getQuoteV3(**args**) internal view returns (uint256
amountOut) {
    //...
    amountIn -= (amountIn * IUniswapV3Pool(poolAddress).fee()) /
MAX_DENOMINATOR;
    //...
}

function _getQuoteV3(**args**) internal view returns (uint256
amountOut) {
    //...
    amountIn -= (amountIn * IUniswapV3Pool(poolAddress).fee()) /
MAX_DENOMINATOR;
    //...
}
```

### Recommendation

It is recommended to account for the possibility that the denominator of the pool may differ from the `MAX_DENOMINATOR`.

## ILU - Inconsistent Library Use

Criticality	Medium
Location	LpRouter.sol#L382,665,670,673
Status	Unresolved

### Description

`_prepareExitPoolRequest` uses the `WeightedPoolUserData` and `StablePoolUserData` according to the case provided. Also in `_prepareJoinPoolRequest` only `StablePoolUserData` is used.

If a pool is actually a weighted pool, the join or exit operation using `StablePoolUserData` might fail or behave unexpectedly because weighted pools typically use `WeightedPoolUserData`.

Conversely, if the pool is a stable pool, the exit operation might incorrectly use `WeightedPoolUserData` according to the case provided, leading to incompatible behavior.

```
request.userData =
abi.encode(StablePoolUserData.JoinKind.EXACT_TOKENS_IN_FOR_BPT_
OUT, newMaxAmountsIn, 0);
//...
request.userData = abi.encode(

WeightedPoolUserData.ExitKind.EXACT_BPT_IN_FOR_ONE_TOKEN_OUT,
    lpAmount,
    tokenOutIndex
);
//...
request.userData =
abi.encode(WeightedPoolUserData.ExitKind.EXACT_BPT_IN_FOR_TOKEN
S_OUT, lpAmount);
//...
request.userData = abi.encode(
    StablePoolUserData.ExitKind.EXACT_BPT_IN_FOR_ONE_TOKEN_OUT,
    lpAmount,
    tokenOutIndex
);
```

## Recommendation

The contract should consistently determine the pool type (stable or weighted) and use the appropriate data encoding for both join and exit operations.

## PRE - Potential Reentrance Exploit

Criticality	Minor / Informative
Location	ZapperBase.sol#L249,278,293
Status	Unresolved

### Description

The contract makes an external call to transfer funds to recipients using the payable transfer method. The recipient could be a malicious contract that has an untrusted code in its fallback function that makes a recursive call back to the original contract. The re-entrance exploit could be used by a malicious user to drain the contract's funds or to perform unauthorized actions. This could happen because the original contract does not update the state before sending funds.

Even if `SwapRouter` and `LpRouter` are protected from reentrancies it would be highly beneficial if zappers are also protected from them, especially since in some cases recipient is a parameter added by the user.

```
function _returnAssets(
    address[] memory tokens,
    address recipient
) internal sphereXGuardInternal(0x65afacc1) returns
(ReturnedAsset[] memory returnedAssets) {
    //...
    (bool success, ) = recipient.call{value: balance}(new
bytes(0));
    if (!success) revert ETHTransferFailed();
    } else {
        //...
    }
}

function _returnAsset(
    address token,
    address recipient
) internal sphereXGuardInternal(0xf45372f8) returns
(ReturnedAsset[] memory returnedAssets) {
    //...
    returnedAssets = _returnAssets(tokens, recipient);
}

function _returnAssetWithAmount(address token, address
recipient, uint256 amount) internal {
    //...
    (bool success, ) = recipient.call{value: amount}(new
bytes(0));
    if (!success) revert ETHTransferFailed();
    //...
}
```

## Recommendation

The team is advised to prevent the potential re-entrance exploit as part of the solidity best practices. Some suggestions are:

- Add lockers/mutexes in the method scope. It is important to note that mutexes do not prevent cross-function reentrancy attacks.
- Do Not allow contract addresses to receive funds.
- Proceed with the external call as the last statement of the method, so that the state will have been updated properly during the re-entrance phase.



## CCR - Contract Centralization Risk

<b>Criticality</b>	Minor / Informative
<b>Location</b>	controllers/Controller.sol#L154,163,172,217 vaults/Vault.sol#L140,151,162,173,191,202 StrategyFactoryBase.sol#L76,85,94,103,264 ZapperBase.sol#L117,127,138,149,159,170,181 InfraredZapper.sol#L155 LpRouter.sol#L75,83,92,101 SwapRouter.sol#L121,129,137,146,156,174
<b>Status</b>	Unresolved

### Description

The contract's functionality and behavior are heavily dependent on external parameters or configurations. While external configuration can offer flexibility, it also poses several centralization risks that warrant attention. Centralization risks arising from the dependence on external configuration include Single Point of Control, Vulnerability to Attacks, Operational Delays, Trust Dependencies, and Decentralization Erosion.

```
function setGovernance(address governanceAddress) public
onlyGovernance sphereXGuardPublic(0xefc74f14, 0xab033ea9) {}

function setTimelock(address timelockAddress) public
onlyTimelock sphereXGuardPublic(0xdecf35e6, 0xbdacb303) {}

function setController(address controllerAddress) external
onlyTimelock sphereXGuardExternal(0x431250ca) {}

function setDepositFee(uint16 newFee) external onlyGovernance
{}

function setWithdrawFee(uint16 newFee) external
onlyGovernance {}

function setFeeRecipient(address feeRecipientAddress)
external onlyGovernance sphereXGuardExternal(0x4fdebcf1) {}

function setStrategist(address strategistAddress) public
onlyGovernance sphereXGuardPublic(0x15f23c15, 0xc7b9d530) {}

function setStrategy(
    address asset,
    address strategy
) public nonReentrant onlyStrategist
sphereXGuardPublic(0x632853be, 0x72cb5d97) {}

function setMin(uint256 minRatio) external onlyGovernance
sphereXGuardExternal(0xdbf26102) {}

function setSphereXEngine(address sphereXEngineAddress)
external onlyDev {}

function setVaultFactory(address factoryAddress) external
onlyDev {}

function setControllerFactory(address factoryAddress)
external onlyDev {}

function createVault(
    VaultCreateParams calldata params
) external onlyDev onlyNewAsset(params.asset) returns (IVault
vault, IController controller, IStrategy strategy) {}

function setDev(address devAddress) external onlyDev {}

function setSwapRouter(address routerAddress) external
onlyGovernance sphereXGuardExternal(0xd473ef3c) {}

function setBexLpToTokenIn(address bexLp, address tokenIn)
```

```
external onlyGovernance {}

    function setLpRouter(address routerAddress) external
onlyGovernance sphereXGuardExternal(0x26b2eef4) {}

    function setStableCoin(address stablecoinAddress) external
onlyGovernance sphereXGuardExternal(0x5b9fdae4) {}

    function setZapInFee(uint16 newFee) external onlyGovernance
{}

    function setZapOutFee(uint16 newFee) external onlyGovernance
{}

    function setFeeRecipient(address newRecipient) external
onlyGovernance {}

    function setAssetInfo(address asset, bool isSingleToken,
IDexType.DexType dex) external onlyGovernance {}

    function setRouter(uint8 dex, address router) external
onlyGovernance sphereXGuardExternal(0xa3f6b48c) {}

    function setDefaultDex(uint8 dex) external onlyGovernance
sphereXGuardExternal(0x76f61bc5) {}

    function setFactory(uint8 dex, address factory) external
onlyGovernance sphereXGuardExternal(0x93b83854) {}

    function setPool(
        address tokenIn,
        address tokenOut,
        address pool
    ) external onlyGovernance sphereXGuardExternal(0x93b83854) {}

    function setSwapRoute(
        address tokenIn,
        address tokenOut,
        SwapRoutePath[] memory path,
        bool reversePath
    ) external onlyGovernance sphereXGuardExternal(0x1a19f525) {}
```

## Recommendation

To address this finding and mitigate centralization risks, it is recommended to evaluate the feasibility of migrating critical configurations and functionality into the contract's codebase itself. This approach would reduce external dependencies and enhance the contract's self-sufficiency. It is essential to carefully weigh the trade-offs between external configuration flexibility and the risks associated with centralization.

## ELFM - Exceeds Fees Limit

Criticality	Minor / Informative
Location	StrategyBase.sol#L27
Status	Unresolved

### Description

The contract timelock address has the authority to increase performance fees over the allowed limit of 25%. The timelock address may take advantage of it by calling the `setPerformanceDevFee` or `setPerformanceTreasuryFee` function with a high percentage value.

```
uint16 public constant MAX_PERFORMANCE_FEE = 5000;
function setPerformanceDevFee(uint16 fee) external onlyTimelock
sphereXGuardExternal(0x39189706) {
    if (fee + performanceTreasuryFee > MAX_PERFORMANCE_FEE)
revert FeeTooHigh(fee, MAX_PERFORMANCE_FEE);
    uint16 old = performanceDevFee;
    performanceDevFee = fee;
    emit PerformanceDevFeeChanged(old, fee);
}

function setPerformanceTreasuryFee(uint16 fee) external
onlyTimelock sphereXGuardExternal(0x76672d92) {
    if (fee + performanceDevFee > MAX_PERFORMANCE_FEE) revert
FeeTooHigh(fee, MAX_PERFORMANCE_FEE);
    uint16 old = performanceTreasuryFee;
    performanceTreasuryFee = fee;
    emit PerformanceTreasuryFeeChanged(old, fee);
}
```

## Recommendation

The contract could embody a check for the maximum acceptable value. The team should carefully manage the private keys of the timelock's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions.

### Temporary Solutions:

These measurements do not decrease the severity of the finding

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-signature wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

## GRLNS - Gamma Remove Liquidity Not Supported

Criticality	Minor / Informative
Location	LpRouter.sol#L132
Status	Unresolved

### Description

`LpRouter` does not support removing liquidity for the `DexType.GAMMA`. Since `LpRouter` allows for adding liquidity to a `GAMMA` `DexType` there should also be functionality for removing it.

```
function removeLiquidity(
    address lp,
    uint256 lpAmount,
    address recipient,
    address tokenOut,
    DexType dexType
) public override nonReentrant sphereXGuardPublic(0xd04d32ff,
0x31512b67) returns (uint256 tokenOutAmount) {
    address router = routers[uint8(dexType)];
    if (dexType == DexType.BEX) {
        return _removeLiquidityBex(IBeraPool(lp), lpAmount,
recipient, tokenOut);
    } else if (dexType == DexType.STEER) {
        return
_removeLiquiditySteer(ISushiMultiPositionLiquidityManager(lp),
lpAmount, recipient, tokenOut);
    } else if (dexType == DexType.KODIAK_V3) {
        return _removeLiquidityKodiak(IKodiakVaultV1(lp),
lpAmount, recipient, router, tokenOut);
    } else {
        revert UnsupportedDexType();
    }
}
```

### Recommendation

The team should consider adding support for removing liquidity for `DexType.GAMMA`.

## MC - Missing Check

Criticality	Minor / Informative
Location	SwapRouter.sol#L72
Status	Unresolved

### Description

The contract is processing variables that have not been properly sanitized and checked that they form the proper shape. These variables may produce vulnerability issues.

Specifically, a check is missing to ensure that `balancerQueriesAddress` is not `address(0)`

```
balancerQueries = balancerQueriesAddress;
```

### Recommendation

The team is advised to properly check the variables according to the required specifications.



## MN - Misspelled Naming

<b>Criticality</b>	Minor / Informative
<b>Location</b>	SwapRouter.sol#L174
<b>Status</b>	Unresolved

### Description

The contract is designed to manage swap routes through the `setSwapRoute` function. However, within this function, the word "route" is used instead of "router". This misspelling may cause confusion for developers and users interacting with the code, as it deviates from the expected terminology used to describe swap routes. While this does not directly affect the functionality of the contract, it may reduce readability and increase the likelihood of misunderstandings or mistakes during future development or maintenance.

```
function setSwapRoute(  
  address tokenIn,  
  address tokenOut,  
  SwapRoutePath[] memory path,  
  bool reversePath  
) external onlyGovernance sphereXGuardExternal(0x1a19f525) {  
  SwapRoutePath[] storage swapRouteForward =  
  swapRoutes[tokenIn][tokenOut];  
  if (swapRouteForward.length > 0) delete  
  swapRoutes[tokenIn][tokenOut];  
  for (uint256 i = 0; i < path.length; i++) {  
    swapRouteForward.push(path[i]);  
    if (!path[i].isMultiPath && path[i].pool != address(0)) {  
      pools[path[i].tokenIn][path[i].tokenOut] = path[i].pool;  
      pools[path[i].tokenOut][path[i].tokenIn] = path[i].pool;  
    }  
  }  
  if (reversePath) {  
    SwapRoutePath[] storage swapRouteReverse =  
    swapRoutes[tokenOut][tokenIn];  
    if (swapRouteReverse.length > 0) delete  
    swapRoutes[tokenOut][tokenIn];  
    for (uint256 i = 0; i < path.length; i++) {  
      SwapRoutePath memory inversePath = path[path.length - i -  
1];  
      swapRouteReverse.push(  
        SwapRoutePath({  
          tokenIn: inversePath.tokenOut,  
          tokenOut: inversePath.tokenIn,  
          dex: inversePath.dex,  
          isMultiPath: inversePath.isMultiPath,  
          pool: inversePath.pool  
        })  
      );  
    }  
  }  
  emit SetSwapRoute(tokenIn, tokenOut, path, reversePath);  
}
```

## Recommendation

It is recommended to correct the misspelled names to align with the intended term "router". Consistent and accurate naming conventions enhance code readability, maintainability, and the overall clarity of the contract. Adhering to clear naming practices reduces potential misinterpretation by developers and auditors.

## ORA - Overwriting Rewards Amount

Criticality	Minor / Informative
Location	InfraredStrategy.sol#L108
Status	Unresolved

### Description

`harvest` function loops through the `rewardTokensLength` to find a `rewardToken` that is equal to the address of the `asset`. However if multiple `rewardToken` have this address then only the last one will be harvested since `newAssets` is getting overwritten.

```
for (uint256 i = 0; i < rewardTokensLength; i++) {
    address rewardToken = staking.rewardTokens(i);
    uint256 rewardAmount =
    IERC20(rewardToken).balanceOf(address(this));
    if (rewardToken == address(asset)) {
        newAssets = rewardAmount;
    } else if (rewardAmount > 0 && rewardToken != wrappedNative)
    {
        IERC20(rewardToken).safeTransfer(address(swapRouter),
        rewardAmount);
        swapRouter.swapWithDefaultDex(rewardToken, wrappedNative,
        rewardAmount, 0, address(this));
    }
}
```

### Recommendation

The team could consider adding the amount instead of overwriting it. In that case, the `harvest` function will account for all the possible `asset` tokens instead of the last one.

## PLPI - Potential Liquidity Provision Inadequacy

<b>Criticality</b>	Minor / Informative
<b>Location</b>	utils/SwapRouter.sol#L450,518,566,609,643,678
<b>Status</b>	Unresolved

### Description

The contract operates under the assumption that liquidity is consistently provided to the pair between the contract's token and the native currency. However, there is a possibility that liquidity is provided to a different pair. This inadequacy in liquidity provision in the main pair could expose the contract to risks. Specifically, during eligible transactions, where the contract attempts to swap tokens with the main pair, a failure may occur if liquidity has been added to a pair other than the primary one. Consequently, transactions triggering the swap functionality will result in a revert.

```
function _swapWithRoute(
    address tokenIn,
    address tokenOut,
    uint256 amountIn,
    uint256 amountOutMinimum,
    address recipient
) internal sphereXGuardInternal(0xf01e8b19) returns (uint256
amountOut) {}

function _swapBex(
    address tokenIn,
    address tokenOut,
    uint256 amountIn,
    uint256 amountOutMinimum,
    address recipient,
    IBeraPool pool
) internal returns (uint256 amountOut) {}

function _swapV3(
    address tokenIn,
    address tokenOut,
    uint256 amountIn,
    uint256 amountOutMinimum,
    address recipient,
    address router,
    address factory
) internal sphereXGuardInternal(0xd2c5d247) returns (uint256
amountOut) {}

function _swapV3WithPath(
    address[] memory path,
    uint256 amountIn,
    uint256 amountOutMinimum,
    address recipient,
    address router,
    address factory
) internal sphereXGuardInternal(0x7d29fbe4) returns (uint256
amountOut) {}

function _swapV2(
    address tokenIn,
    address tokenOut,
    uint256 amountIn,
    uint256 amountOutMinimum,
    address recipient,
    address router
) internal sphereXGuardInternal(0xd8b71976) returns (uint256
amountOut) {}
```

```
function _swapV2WithPath(  
    address[] memory path,  
    uint256 amountIn,  
    uint256 amountOutMinimum,  
    address recipient,  
    address router  
) internal sphereXGuardInternal(0x0de072e7) returns (uint256  
amountOut) {}
```

## Recommendation

The team is advised to implement a runtime mechanism to check if the pair has adequate liquidity provisions. This feature allows the contract to omit token swaps if the pair does not have adequate liquidity provisions, significantly minimizing the risk of potential failures.

Furthermore, the team could ensure the contract has the capability to switch its active pair in case liquidity is added to another pair.

Additionally, the contract could be designed to tolerate potential reverts from the swap functionality, especially when it is a part of the main transfer flow. This can be achieved by executing the contract's token swaps in a non-reversible manner, thereby ensuring a more resilient and predictable operation.

## SAU - Swapped Amount Uninitialized

Criticality	Minor / Informative
Location	LpRouter.sol#L305
Status	Unresolved

### Description

`handleTokenSwaps` is used to swap `tokenIn` to a token that can be used to provide liquidity to a specified `lp`. However if either of the initial amounts added is zero or when more than two tokens are added as input the entire contract's balance of `tokenIn` is used to calculate the `swappedAmount1`. The `swappedAmount0` is not calculated, therefore it will always be zero.

```
{
    uint256 tokenInBalance =
    IERC20(tokenIn).balanceOf(address(this));
    if (bexLpToTokenIn[lp] != address(0)) {
        token1 = bexLpToTokenIn[lp];
    }
    if (address(tokenIn) != token1) {
        IERC20(tokenIn).safeTransfer(address(swapRouter),
        tokenInBalance);
        swappedAmount1 = swapRouter.swapWithDefaultDex(tokenIn,
        token1, tokenInBalance, 0, address(this));
    } else {
        swappedAmount1 = tokenInBalance;
    }
    IERC20(token1).forceApprove(beraVault, swappedAmount1);
}
```

### Recommendation

The team is advised to consider the situations that can occur when trying to provide liquidity with zero amounts of tokens.

## UBUFL - Unintended Balance Used For Liquidity

Criticality	Minor / Informative
Location	LpRouter.sol#L107
Status	Unresolved

### Description

The contract declares `addLiquidity` as a public function. Additionally, `addLiquidity` does not transfer tokens from the user to the contract but instead assumes that they are provided beforehand. This can create situations where users provide tokens in the contract and the tokens are used before they use `addLiquidity`. Since users are able to choose the `amountIn` they could use any amount of tokens stored in the contract. Additionally, excess tokens will be returned to the caller.

```
function addLiquidity(
    address lp,
    address tokenIn,
    uint256 amountIn,
    address recipient,
    DexType dexType
) public nonReentrant sphereXGuardPublic(0xed92e336,
0xee52e659) returns (uint256 lpAmountOut) {
    //...
}
```



Additionally in `_addLiquidityBex` if `amountIn` is zero or one then `_handleTokenSwaps` will return `amount1` calculated by the entire contract's balance.

```
function _addLiquidityBex(
    IBeraPool lp,
    address tokenIn,
    uint256 amountIn,
    address recipient
) internal sphereXGuardInternal(0x6a9e79f4) returns (uint256
lpAmountOut) {
    //...
    (amount0, amount1) = _handleTokenSwaps(tokenIn,
address(lp), tokens, amount0, amount1);
    //...
}

function _handleTokenSwaps(
    address tokenIn,
    address lp,
    IERC20[] memory tokens,
    uint256 amount0,
    uint256 amount1
) internal returns (uint256 swappedAmount0, uint256
swappedAmount1) {
    if (amount0 > 0 && amount1 > 0) {
        //...
    } else {
        uint256 tokenInBalance =
IERC20(tokenIn).balanceOf(address(this));
        //...
    }
}
```

## Recommendation

The team could consider using a strategy that allows users to first approve the tokens to the `LpRouter` and then use `addLiquidity` to utilize only the approved amount.

## UTPD - Unverified Third Party Dependencies

Criticality	Minor / Informative
Location	strategies/StrategyBase.sol#L192 strategies/InfraredStrategy.sol#L79 StrategyFactoryBase.sol#L168 LpRouter.sol#L536,584
Status	Unresolved

### Description

The contract uses an external contract in order to determine the transaction's flow. The external contract is untrusted. As a result, it may produce security issues and harm the transactions.

```
function _swapBGTToAsset() internal returns (uint256 amount) {
    uint256 balance = IERC20(bgt).balanceOf(address(this));
    if (balance > 0) {
        bgt.redeem(address(this), balance);
        ...
    }
    ...
}
```

```
function deposit() public override {
    ...
    staking.stake(balance);
}
```

```
address public sphereXEngine;
...
ISphereXEngine(sphereXEngine).addAllowedSenderOnChain(address(controller));
ISphereXEngine(sphereXEngine).addAllowedSenderOnChain(address(vault));
};
```

```
...  
ISteerPeriphery(router).deposit(address(lp),  
liquidityInfo.amount0, liquidityInfo.amount1, 0, 0,  
address(this));
```

## Recommendation

The contract should use a trusted external source. A trusted source could be either a commonly recognized or an audited contract. The pointing addresses should not be able to change after the initialization.

## L04 - Conformance to Solidity Naming Conventions

<b>Criticality</b>	Minor / Informative
<b>Location</b>	ArberaStrategy.sol#L128
<b>Status</b>	Unresolved

### Description

The Solidity style guide is a set of guidelines for writing clean and consistent Solidity code. Adhering to a style guide can help improve the readability and maintainability of the Solidity code, making it easier for others to understand and work with.

The followings are a few key points from the Solidity style guide:

1. Use camelCase for function and variable names, with the first letter in lowercase (e.g., myVariable, updateCounter).
2. Use PascalCase for contract, struct, and enum names, with the first letter in uppercase (e.g., MyContract, UserStruct, ErrorEnum).
3. Use uppercase for constant variables and enums (e.g., MAX\_VALUE, ERROR\_CODE).
4. Use indentation to improve readability and structure.
5. Use spaces between operators and after commas.
6. Use comments to explain the purpose and behavior of the code.
7. Keep lines short (around 120 characters) to improve readability.

```
address _rewardToken
```

### Recommendation

By following the Solidity naming convention guidelines, the codebase increased the readability, maintainability, and makes it easier to work with.

Find more information on the Solidity documentation

<https://docs.soliditylang.org/en/stable/style-guide.html#naming-conventions>.

## L09 - Dead Code Elimination

Criticality	Minor / Informative
Location	ZapperBase.sol#L203,232 LpRouter.sol#L163
Status	Unresolved

### Description

In Solidity, dead code is code that is written in the contract, but is never executed or reached during normal contract execution. Dead code can occur for a variety of reasons, such as:

- Conditional statements that are always false.
- Functions that are never called.
- Unreachable code (e.g., code that follows a return statement).

Dead code can make a contract more difficult to understand and maintain, and can also increase the size of the contract and the cost of deploying and interacting with it.

```
function _approveTokenIfNeeded(
    address tokenAddress,
    address spenderAddress
) internal sphereXGuardInternal(0x367a5b5d) {
    if (IERC20(tokenAddress).allowance(address(this),
spenderAddress) == 0) {
        IERC20(tokenAddress).approve(spenderAddress,
type(uint256).max);
    }
}
...

function _returnAssets(address[] memory tokens) internal {
    uint256 balance;
    for (uint256 i = 0; i < tokens.length; i++) {
        if (tokens[i] == address(0)) continue;
        balance = IERC20(tokens[i]).balanceOf(address(this));
        if (balance > 0) {
            IERC20(tokens[i]).safeTransfer(msg.sender, balance);
        }
    }
}
```

## Recommendation

To avoid creating dead code, it's important to carefully consider the logic and flow of the contract and to remove any code that is not needed or that is never executed. This can help improve the clarity and efficiency of the contract.

## L14 - Uninitialized Variables in Local Scope

Criticality	Minor / Informative
Location	LpRouter.sol#L397,398,451,507,550,597,613 ArberaZapper.sol#L516
Status	Unresolved

### Description

Using an uninitialized local variable can lead to unpredictable behavior and potentially cause errors in the contract. It's important to always initialize local variables with appropriate values before using them.

```
uint256 amount0  
uint256 amount1  
LiquidityAddInfo memory liquidityInfo  
LiquidityRemoveInfo memory liquidityInfo  
uint256 assetsIn
```

### Recommendation

By initializing local variables before using them, the contract ensures that the functions behave as expected and avoid potential issues.

## L15 - Local Scope Variable Shadowing

<b>Criticality</b>	Minor / Informative
<b>Location</b>	SteerZapper.sol#L31,32,33,34,35,36,37 KodiakZapper.sol#L31,33,34,35,36,37 InfraredZapper.sol#L27,28,29,30,31,32,33 ArberaZapper.sol#L26,27,28,29,30,31,32
<b>Status</b>	Unresolved

### Description

Local scope variable shadowing occurs when a local variable with the same name as a variable in an outer scope is declared within a function or code block. When this happens, the local variable "shadows" the outer variable, meaning that it takes precedence over the outer variable within the scope in which it is declared.

```
address wrappedNative
address stablecoin
address swapRouter
address lpRouter
address feeRecipient
uint16 zapInFee
uint16 zapOutFee
address strategist
address governance
address timelock
```

### Recommendation

It's important to be aware of shadowing when working with local variables, as it can lead to confusion and unintended consequences if not used correctly. It's generally a good idea to choose unique names for local variables to avoid shadowing outer variables and causing confusion.



## L17 - Usage of Solidity Assembly

Criticality	Minor / Informative
Location	StrategyFactoryBase.sol#L309 StrategyBase.sol#L437
Status	Unresolved

### Description

Using assembly can be useful for optimizing code, but it can also be error-prone. It's important to carefully test and debug assembly code to ensure that it is correct and does not contain any errors.

Some common types of errors that can occur when using assembly in Solidity include Syntax, Type, Out-of-bounds, Stack, and Revert.

```
assembly {  
    strategyAddress := create(0, add(bytecode, 0x20),  
mload(bytecode))  
  
    if iszero(extcodesize(strategyAddress)) {  
        revert(0, 0)  
    }  
}  
...
```

### Recommendation

It is recommended to use assembly sparingly and only when necessary, as it can be difficult to read and understand compared to Solidity code.

## Functions Analysis

Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
<b>ZapperBase</b>	Implementation	SphereXProtected, ReentrancyGuard, IZapper		
		Public	✓	-
	setGovernance	External	✓	onlyGovernance
	setSwapRouter	External	✓	onlyGovernance sphereXGuardExternal
	setLpRouter	External	✓	onlyGovernance sphereXGuardExternal
	setStableCoin	External	✓	onlyGovernance sphereXGuardExternal
	setZapInFee	External	✓	onlyGovernance
	setZapOutFee	External	✓	onlyGovernance
	setFeeRecipient	External	✓	onlyGovernance
		External	Payable	-
	_revertAddressZero	Internal		
	_approveTokenIfNeeded	Internal	✓	sphereXGuardInternal
	_safeTransferFromTokens	Internal	✓	sphereXGuardInternal
	_divideAmountInRatio	Internal		

	_returnAssets	Internal	✓	sphereXGuardInternal
	_returnAsset	Internal	✓	sphereXGuardInternal
	_returnAssetWithAmount	Internal	✓	
	_transferFee	Internal	✓	
	swapToAssets	Public	✓	-
	swapFromAssets	Public	✓	-
	zapIn	External	Payable	nonReentrant sphereXGuardExternal
	zapOut	External	✓	nonReentrant sphereXGuardExternal
<b>VaultFactory</b>	Implementation	SphereXProtectedBase, IVaultFactory		
		Public	✓	SphereXProtectedBase
	_revertAddressZero	Internal		
	setDev	External	✓	onlyDev sphereXGuardExternal
	setWhitelistedStrategyFactory	External	✓	onlyDev sphereXGuardExternal
	setSphereXProtected	Private	✓	sphereXGuardInternal
	createVault	External	✓	onlyWhitelisted StrategyFactory sphereXGuardExternal
<b>Vault</b>	Implementation	SphereXProtected, ReentrancyGuard,		

		ERC4626, IVault		
		Public	✓	ERC4626 ERC20
	_revertOnlyGovernance	Internal		
	_revertOnlyTimelock	Internal		
	_revertOnlyController	Internal		
	_revertAddressZero	Internal		
	decimals	Public		-
	setMin	External	✓	onlyGovernanc e sphereXGuardE xternal
	setDepositFee	External	✓	onlyGovernanc e
	setWithdrawFee	External	✓	onlyGovernanc e
	setGovernance	External	✓	onlyGovernanc e sphereXGuardE xternal
	setFeeRecipient	External	✓	onlyGovernanc e sphereXGuardE xternal
	setTimelock	External	✓	onlyTimelock sphereXGuardE xternal
	setController	External	✓	onlyTimelock sphereXGuardE xternal
	available	Public		-
	totalAssets	Public		-
	earn	Public	✓	nonReentrant sphereXGuardP ublic
	harvest	External	✓	nonReentrant onlyController

				sphereXGuardExternal
	_withdraw	Internal	✓	sphereXGuardInternal
	_deposit	Internal	✓	sphereXGuardInternal
	deposit	Public	✓	nonReentrant sphereXGuardPublic
	redeem	Public	✓	nonReentrant sphereXGuardPublic
<b>SwapRouter</b>	Implementation	SphereXProtected, ReentrancyGuard, ISwapRouter		
		Public	✓	-
	validateSwapParams	Internal		
	validateSwapWithPathParams	Internal		
	setGovernance	Public	✓	onlyGovernance sphereXGuardPublic
	setDefaultDex	External	✓	onlyGovernance sphereXGuardExternal
	setRouter	External	✓	onlyGovernance sphereXGuardExternal
	setFactory	External	✓	onlyGovernance sphereXGuardExternal
	setPool	External	✓	onlyGovernance sphereXGuardExternal
	setSwapRoute	External	✓	onlyGovernance

				sphereXGuardExternal
	swapWithDefaultDex	External	✓	resetPathLength sphereXGuardExternal
	swap	Public	✓	nonReentrant sphereXGuardPublic
	swapWithPathWithDefaultDex	Public	✓	nonReentrant sphereXGuardPublic
	swapWithPath	Public	✓	nonReentrant sphereXGuardPublic
	getQuoteWithDefaultDex	External	✓	-
	getQuote	Public	✓	-
	getQuoteWithPathWithDefaultDex	External		-
	getQuoteWithPath	Public		-
	_revertAddressZero	Internal		
	_revertZeroAmount	Internal		
	_revertInvalidPathLength	Internal		
	_findMostLiquidV3Pool	Internal		
	_swapWithRoute	Internal	✓	sphereXGuardInternal
	_swapBex	Internal	✓	
	_swapV3	Internal	✓	sphereXGuardInternal
	_swapV3WithPath	Internal	✓	sphereXGuardInternal
	_swapV2	Internal	✓	sphereXGuardInternal
	_swapV2WithPath	Internal	✓	sphereXGuardInternal
	_getQuoteV3	Internal		

	_getQuoteV3WithPath	Internal		
	_getBexQuote	Internal	✓	
	_getQuoteV2	Internal		
	_getQuoteV2WithPath	Internal		
<b>StrategyFactoryBase</b>	Implementation	IStrategyFactory		
		Public	✓	-
	revertOnlyDev	Private		
	_revertAddressZero	Internal		
	setDev	External	✓	onlyDev
	setSphereXEngine	External	✓	onlyDev
	setVaultFactory	External	✓	onlyDev
	setControllerFactory	External	✓	onlyDev
	_setAddressAsSpherexProtected	Internal	✓	
	_createControllerAndVault	Internal	✓	
	_setupVault	Internal	✓	
	revertIfNonInitializedParams	Private		
	_encodeParamsAndController	Private		
	createVault	External	✓	onlyDev onlyNewAsset
	_deployStrategyByteCode	Internal	✓	
<b>StrategyBase</b>	Implementation	SphereXProtected, ReentrancyGuard, IStrategy		
		Public	✓	-

		External	Payable	-
	balanceOfAsset	Public		-
	balanceOfPool	Public		-
	balanceOf	Public		-
	_revertAddressZero	Internal		
	_revertOnlyGovernance	Internal		
	_revertOnlyTimelock	Internal		
	_revertOnlyController	Internal		
	_revertOnlyBenevolent	Internal		
	_swapBGTToAsset	Internal	✓	sphereXGuardInternal
	whitelistHarvester	External	✓	sphereXGuardExternal onlyBenevolent
	revokeHarvester	External	✓	sphereXGuardExternal onlyBenevolent
	setWithdrawalDevFundFee	External	✓	onlyTimelock sphereXGuardExternal
	setWithdrawalTreasuryFee	External	✓	onlyTimelock sphereXGuardExternal
	setPerformanceDevFee	External	✓	onlyTimelock sphereXGuardExternal
	setPerformanceTreasuryFee	External	✓	onlyTimelock sphereXGuardExternal
	setStrategist	External	✓	onlyGovernance sphereXGuardExternal
	setGovernance	External	✓	onlyGovernance sphereXGuardExternal



	setTimelock	External	✓	onlyTimelock sphereXGuardE xternal
	setController	External	✓	onlyTimelock sphereXGuardE xternal
	setSwapRouter	External	✓	onlyGovernanc e sphereXGuardE xternal
	setLpRouter	External	✓	onlyGovernanc e sphereXGuardE xternal
	setZapper	External	✓	onlyGovernanc e sphereXGuardE xternal
	deposit	Public	✓	-
	getHarvestable	External		-
	harvest	Public	✓	nonReentrant onlyBenevolent sphereXGuardP ublic
	_withdrawSome	Internal	✓	
	withdraw	External	✓	nonReentrant onlyController sphereXGuardE xternal
	withdraw	External	✓	nonReentrant onlyController sphereXGuardE xternal
	withdrawForSwap	External	✓	nonReentrant onlyController sphereXGuardE xternal
	withdrawAll	External	✓	nonReentrant onlyController sphereXGuardE xternal
	_withdrawAll	Internal	✓	sphereXGuardI nternal

	execute	Public	Payable	onlyTimelock sphereXGuardP ublic
	_distributePerformanceFeesBasedAmou ntAndDeposit	Internal	✓	sphereXGuardI nternal
<b>SteerZapper</b>	Implementation	ZapperBase		
		Public	✓	ZapperBase
	swapToAssets	Public	✓	sphereXGuardP ublic
	swapFromAssets	Public	✓	sphereXGuardP ublic
<b>SteerStrategy</b>	Implementation	StrategyBas e		
		Public	✓	StrategyBase
	deposit	Public	✓	sphereXGuardP ublic
	balanceOfPool	Public		-
	_withdrawSome	Internal	✓	sphereXGuardI nternal
<b>SteerFactory</b>	Implementation	StrategyFact oryBase		
		Public	✓	StrategyFactory Base
	createVaultWithParams	External	✓	onlyDev onlyNewAsset
<b>LpRouter</b>	Implementation	SphereXProt ected, ReentrancyG uard, ILpRouter		
		Public	✓	-

	setGovernance	Public	✓	onlyGovernance sphereXGuardPublic
	setSwapRouter	External	✓	onlyGovernance sphereXGuardExternal
	setBexLpToTokenIn	External	✓	onlyGovernance
	setRouter	External	✓	onlyGovernance sphereXGuardExternal
	addLiquidity	Public	✓	nonReentrant sphereXGuardPublic
	removeLiquidity	Public	✓	nonReentrant sphereXGuardPublic
	_revertAddressZero	Internal		
	_returnAssets	Internal	✓	
	_swapAndReturnAssets	Internal	✓	
	_swapAndReturnAssets	Internal	✓	
	_approveTokenIfNeeded	Internal	✓	sphereXGuardInternal
	_divideAmountInRatio	Internal		
	_getAmountsForLiquidityInRatio	Internal	✓	
	_handleTokenSwaps	Internal	✓	
	_prepareJoinPoolRequest	Internal		
	_addLiquidityBex	Internal	✓	sphereXGuardInternal
	_swapTokensForLiquidity	Internal	✓	
	_addLiquidityKodiak	Internal	✓	
	_addLiquiditySteer	Internal	✓	

	_addLiquidityGamma	Internal	✓	
	_removeLiquiditySteer	Internal	✓	
	_removeLiquidityKodiak	Internal	✓	
	_prepareExitPoolRequest	Internal		
	_removeLiquidityBex	Internal	✓	sphereXGuardInternal
<b>KodiakZapper</b>	Implementation	ZapperBase		
		Public	✓	ZapperBase
	swapToAssets	Public	✓	sphereXGuardPublic
	swapFromAssets	Public	✓	sphereXGuardPublic
<b>KodiakStrategy</b>	Implementation	StrategyBase		
		Public	✓	StrategyBase
	deposit	Public	✓	sphereXGuardPublic
	balanceOfPool	Public		-
	_withdrawSome	Internal	✓	sphereXGuardInternal
<b>KodiakFactory</b>	Implementation	StrategyFactoryBase		
		Public	✓	StrategyFactoryBase
	createVaultWithParams	External	✓	onlyDev onlyNewAsset
<b>InfraredZapper</b>	Implementation	ZapperBase		

		Public	✓	ZapperBase
	swapToAssets	Public	✓	sphereXGuardPublic
	_swapToAssetsLp	Internal	✓	
	swapFromAssets	Public	✓	sphereXGuardPublic
	_swapFromAssetsLp	Internal	✓	
	setAssetInfo	External	✓	onlyGovernance
<b>InfraredStrategy</b>	Implementation	StrategyBase		
		Public	✓	StrategyBase
	deposit	Public	✓	nonReentrant sphereXGuardPublic
	getHarvestable	External		-
	harvest	Public	✓	onlyBenevolent sphereXGuardPublic
	balanceOfPool	Public		-
	_withdrawSome	Internal	✓	sphereXGuardInternal
	setRewardTokensLength	External	✓	onlyGovernance sphereXGuardExternal
	setStaking	External	✓	onlyGovernance
<b>InfraredFactory</b>	Implementation	StrategyFactoryBase		
		Public	✓	StrategyFactoryBase
	deployStrategyWithParamsAndController	Internal	✓	

	createVaultWithParams	External	✓	onlyDev onlyNewAsset
<b>ControllerFactory</b>	Implementation	SphereXProtectedBase, IControllerFactory		
		Public	✓	SphereXProtectedBase
	_revertAddressZero	Internal		
	setDev	External	✓	onlyDev sphereXGuardExternal
	setWhitelistedStrategyFactory	External	✓	onlyDev sphereXGuardExternal
	setSphereXProtected	Private	✓	sphereXGuardInternal
	createController	External	✓	onlyWhitelisted StrategyFactory sphereXGuardExternal
<b>Controller</b>	Implementation	SphereXProtected, ReentrancyGuard, IController		
		Public	✓	-
	_revertAddressZero	Internal		
	_revertOneAddressZero	Internal		
	_revertOnlyGovernance	Internal		
	_revertOnlyStrategist	Internal		
	_revertOnlyTimelock	Internal		
	setDevFund	Public	✓	onlyGovernance sphereXGuardPublic

	setTreasury	Public	✓	onlyGovernance sphereXGuardPublic
	setStrategist	Public	✓	onlyGovernance sphereXGuardPublic
	setGovernance	Public	✓	onlyGovernance sphereXGuardPublic
	setTimelock	Public	✓	onlyTimelock sphereXGuardPublic
	setVault	Public	✓	onlyStrategist sphereXGuardPublic
	approveStrategy	Public	✓	onlyTimelock sphereXGuardPublic
	revokeStrategy	Public	✓	onlyGovernance sphereXGuardPublic
	setStrategy	Public	✓	nonReentrant onlyStrategist sphereXGuardPublic
	earn	Public	✓	nonReentrant sphereXGuardPublic
	balanceOf	External		-
	withdrawAll	Public	✓	nonReentrant onlyStrategist sphereXGuardPublic
	inCaseTokensGetStuck	Public	✓	onlyGovernance sphereXGuardPublic
	inCaseStrategyTokenGetStuck	Public	✓	onlyGovernance sphereXGuardPublic

	withdraw	Public	✓	nonReentrant onlyVault sphereXGuardP ublic
<b>ArberaZapper</b>	Implementation	ZapperBase		
		Public	✓	ZapperBase
	swapToAssets	Public	✓	sphereXGuardP ublic
	swapFromAssets	Public	✓	sphereXGuardP ublic
	swapToAssetsWithBond	Public	✓	sphereXGuardP ublic
	swapFromAssetsWithBond	Public	✓	sphereXGuardP ublic
	_getAmounts	Internal		
	zapIn	External	Payable	nonReentrant sphereXGuardE xternal
	zapInWithBond	External	Payable	nonReentrant sphereXGuardE xternal
	zapOutWithBond	External	✓	nonReentrant sphereXGuardE xternal
<b>ArberaStrategy</b>	Implementation	StrategyBase		
		Public	✓	StrategyBase
	deposit	Public	✓	sphereXGuardP ublic
	getHarvestable	External		-
	harvest	Public	✓	onlyBenevolent sphereXGuardP ublic
	balanceOfPool	Public		-




	_withdrawSome	Internal	✓	sphereXGuardInternal
	setRewardToken	External	✓	onlyGovernance
<b>ArberaFactory</b>	Implementation	StrategyFactoryBase		
		Public	✓	StrategyFactoryBase
	createVaultWithParams	External	✓	onlyDev onlyNewAsset
<b>IVaultFactory</b>	Interface			
	createVault	External	✓	-
<b>IVault</b>	Interface	IERC4626		
	earn	External	✓	-
	available	External		-
	deposit	External	✓	-
	redeem	External	✓	-
<b>ISwapRouter</b>	Interface	IDexType		
	wrappedNative	External		-
	routers	External		-
	factories	External		-
	defaultDex	External		-
	swapWithDefaultDex	External	✓	-
	swap	External	✓	-

	swapWithPathWithDefaultDex	External	✓	-
	swapWithPath	External	✓	-
	getQuoteWithDefaultDex	External	✓	-
	getQuote	External	✓	-
	getQuoteWithPathWithDefaultDex	External		-
	getQuoteWithPath	External		-
<b>ILpRouter</b>	Interface	IDexType		
	addLiquidity	External	✓	-
	removeLiquidity	External	✓	-
<b>IControllerFactory</b>	Interface			
	createController	External	✓	-
<b>IController</b>	Interface			
	treasury	External		-
	devfund	External		-
	strategist	External		-
	governance	External		-
	timelock	External		-
	vaults	External		-
	strategies	External		-
	approvedStrategies	External		-
	balanceOf	External		-

	withdraw	External	✓	-
	earn	External	✓	-
	setVault	External	✓	-
	approveStrategy	External	✓	-
	revokeStrategy	External	✓	-
	setStrategy	External	✓	-
	setStrategist	External	✓	-
	setGovernance	External	✓	-
	setTimelock	External	✓	-

## Inheritance Graph

For the detailed graph file, please refer to the following link:

 [inheritance\\_graph.png](#)

## Summary

The BeraTrax protocol implements a modular and secure yield-generation mechanism through its core components, Vaults, Controllers, Strategies, Factories, and Zappers. This audit investigates security vulnerabilities, evaluates business logic consistency, and identifies potential improvements to enhance system robustness and operational efficiency.

## Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope's prior written consent. This report is not nor should be considered an "endorsement" or "disapproval" of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security. Cyberscope's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

# About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.



**The Cyberscope team**

[cyberscope.io](https://cyberscope.io)