SHERLOCK

Security Review For Spectra Finance

Collaborative Audit Prepared For: **Spectra Finance**
Lead Security Expert(s): **0x73696d616f**
hildingr
Date Audited: **December 16 - December 18, 2025**

Introduction

Metavaults allow a curator to perform actions on top of the Spectra Protocol. This is done by using a composable stack of Zodiac modules. Each module provides a different set of rules and permissions for targeted contracts and functions. In this audit extension, we audit the Bridge Module, which allows the curator to bridge assets across chains with predefined and supervisable paths.

Scope

Repository: perspectivefi/spectra-metavault

Audited Commit: d5ae3dd3fdd7ae7c2c840e12b9dc31c94124f75d

Final Commit: 09de49a70c5d52a8487ce4c7891d58ca1b950456

Files:

- src/gatekeepers/BridgeGatekeeper.sol
- src/MetavaultsRegistry.sol
- src/zaps/BaseBridgeInterface.sol
- src/zaps/BridgeCCTP.sol
- src/zaps/BridgeDeBridge.sol

Final Commit Hash

09de49a70c5d52a8487ce4c7891d58ca1b950456

Findings

Each issue has an assigned severity:

- High issues are directly exploitable security vulnerabilities that need to be fixed.
- Medium issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
- Low/Info issues are non-exploitable, informational findings that do not pose a security risk or impact the system's integrity. These issues are typically cosmetic or related to compliance requirements, and are not considered a priority for remediation.

Issues Found

High	Medium	Low/Info
0	1	11

Issues Not Fixed and Not Acknowledged

High	Medium	Low/Info
0	0	0

Issue M-1: Curator can set destinationCaller to any value which could lead to loss of funds. [RESOLVED]

Source: https://github.com/sherlock-audit/2025-12-spectra-metavault-update-dec-16th/issues/9

Summary

The curator can set `destinationCaller` to any value. This could either through a user error or due to a malicious curator lead to loss of funds as this will be the only address that can release the funds on the destination chain.

Vulnerability Detail

`bridgeAssetCctp()` and `bridgeAsset()` can be called by the curator with any value on the `destinationCaller` parameters.

This will be passed into the CCPT messenger function call

https://github.com/sherlock-audit/2025-12-spectra-metavault-update-dec-16th/blob/bfef1d6c12c1888552eb58e08b95e2d318d75c17/spectra-metavault/src/zaps/BridgeCCTP.sol#L115-L123

On the destination chain `MessageTransmitterV2`will revert as long as the `msg.sender` does not correspond to the `destinationCaller`

https://github.com/circlefin/evm-cctp-contracts/blob/4061786a5726bc05f99fcdb53b0985599f0dbaf7/src/v2/MessageTransmitterV2.sol#L301-L306

Impact

User error: Can freeze the funds if the curator sets this to an address they do not control.

Attack: The curator will be the only actor that can release these funds, even if the curator is removed. The curator could for example set the `destinationCaller` to a non-upgradable smart contract that will only release the funds if X% amount is first deposited into the contract to blackmail users.

Recommendation

Hard code `destinationCaller` to 0 such that the release of the funds to the vault can be triggered by anybody.

Discussion

petarcalic99

Agreed. This is problmematic as there is no way to cancel the bridge action. Will fix by hardcoding the value to 0x so that anyone can finalize the message.

Issue L-1: Missing check making sure msg.value is 0 when bridgeAsset() is called [RESOLVED]

Source: https://github.com/sherlock-audit/2025-12-spectra-metavault-update-dec-16th/issues/6

Summary

The `bridgeAsset()` function in `BridgeCCPT` follows `BaseBridgeInterface` interface and therefor must be `payable` but the CCPT bridge will never utilize any native tokens. If the function was somehow called with `msg.value !=0` the funds would be stuck in the bridge.

https://github.com/sherlock-audit/2025-12-spectra-metavault-update-dec-16th/blob/bfef1d6c12c1888552eb58e08b95e2d318d75c17/spectra-metavault/src/zaps/BridgeCCTP.sol#L45

Recommendation

Add a check to make sure `msg.value==0`.

Discussion

petarcalic99

Agreed, added the check

Issue L-2: Consider adding functionality to patch and cancel deBridge orders [ACKNOWLEDGED]

Source: https://github.com/sherlock-audit/2025-12-spectra-metavault-update-dec-16th/issues/7

This issue has been acknowledged by the team but won't be fixed at this time.

Summary

DeBridge allows for patching and cancelling off messages on the destination chain. By setting the `givePatchAuthoritySrc`, `orderAuthorityAddressDst` and the `allowCancelBeneficiarySrc` in the `Order`.

https://github.com/sherlock-audit/2025-12-spectra-metavault-update-dec-16th/blob/bfef1d6c12c1888552eb58e08b95e2d318d75c17/spectra-metavault/src/zaps/BridgeDeBridge.sol#L179-L191

There is currently no zap functionality for these actions.

Cancelling the order could in an extreme edge case be used to protect users' funds if for some reason the token on the destination chain is de-pegging from the true market-value as the funds are being bridged over.

Recommendation

Consider adding zap functionality for these actions to allow the curator to conveniently adjust the order if necessary.

Discussion

petarcalic99

I agree this function could be usefull and we can add it but on the scripts side, its not necessary to add it in the zap. We just want the curator to be able to cancel his order if needed through the default role modifier.

petarcalic99

For now it is non priority and the scripts are not in scope of the audit so we will add it once the audit is finished. As for the patching functions the curator could execute those through the delay custom call as it should be rare that he needs to perform this action

Issue L-3: Add check on minFinalityThreshold as Off-chain attestation service could enforce longer thresholds [ACKNOWLEDGED]

Source: https://github.com/sherlock-audit/2025-12-spectra-metavault-update-dec-16th/issues/8

This issue has been acknowledged by the team but won't be fixed at this time.

Summary

Based on the following

> In CCTP, each message specifies a `minFinalityThreshold`. This threshold indicates the minimum level of confirmation required for Circle's attestation service (Iris) to attest to the message. Iris will not attest to a message at a confirmation level below the specified minimum threshold. This allows applications to enforce a desired level of finality before acting on an attestation on the destination chain.

Information in circles docs It is unclear if they enforce larger values than 2000 in their off-chain attestation service.

If a curator mistakenly inputs a very large number the message could be stuck until Circle manually attests.

https://github.com/sherlock-audit/2025-12-spectra-metavault-update-dec-16th/blob/bfef1d6c12c1888552eb58e08b95e2d318d75c17/spectra-metavault/src/zaps/BridgeCCTP.sol#L115-L123

Recommendation

Check that `minFinalityThreshold=<2000`

Discussion

petarcalic99

For now the app will fix this parameter and there is a delay associated with the de bridge functions as they carry parameter risks (slippage etc..). So this parameter will be noticed in case the curator goes around our app and sets a different value. Plus for now setting a higher value should be the same as 2000 for now. If not we wish to leave room to set a custom finality to be future proof and not need to upgrade contracts.

https://developers.circle.com/cctp/technical-guide
allowbreak #finality-thresholds "Only two finality thresholds are supported. Any

minFinalityThreshold value below 1000 is treated as 1000, and any value above 1000 is treated as 2000."

Issue L-4: nativeFeeProvided < totalNativeRequired check in BridgeDeBridge::bridgeDeBridgeWithFee is incorrect [RESOLVED]

Source: https://github.com/sherlock-audit/2025-12-spectra-metavault-update-dec-16th/issues/10

Summary

The `nativeFeeProvided < totalNativeRequired` check should be `!=`.

Vulnerability Detail

The source bridge enforces that the native fee provided is exactly the amount + fee, link. Hence, this check is not accurate.

```
if (_orderCreation.giveTokenAddress == address(0)) {
    if (msg.value != _order.giveAmount + globalFixedNativeFee) revert
    ↪  MismatchNativeGiveAmount();
}
```

Impact

There is no impact other than some extra gas cost and readability, since the call will revert later.

Code Snippet

https://github.com/sherlock-audit/2025-12-spectra-metavault-update-dec-16th/pull/5/files#diff-eba31f672a0e150d33e98354d7cada4c4eb270fda831d5b53382b7eaae395260R166

Tool Used

Manual Review

Recommendation

Change the check to be `!=`.

Discussion

petarcalic99

Agreed, we will fix this revert condition

Issue L-5: Struct order in MetavaultsRegistry could be switched for better readability [RESOLVED]

Source: https://github.com/sherlock-audit/2025-12-spectra-metavault-update-dec-16th/issues/11

Summary

The structs `MetavaultsRegistryState` and `MarketInfosState` appear in different order compared to the functions and constants.

Vulnerability Detail

```
// keccak256(abi.encode(uint256(keccak256("spectra.metavault.markets")) - 1)) &
↪  ~bytes32(uint256(0xff))
bytes32 private constant MARKET_INFOS_LOCATION =
    0xed37d1382a6bfbb25f83b33532c043812221900 08afbfed9374fd7204ef8d200;

// keccak256(abi.encode(uint256(keccak256("spectra.metavault.configs")) - 1)) &
↪  ~bytes32(uint256(0xff))
bytes32 private constant MV_CONFIGS_LOCATION =
    0xc529114c4cac99852bfd8483ed99a3cf970cf4610e078f8f8ea71d2fe61b1c00;

// keccak256(abi.encode(uint256(keccak256("spectra.metavault.bridge")) - 1)) &
↪  ~bytes32(uint256(0xff))
bytes32 private constant BRIDGE_CONFIGS_LOCATION =
    0xa3601e533f2d68c8147877ed38f63d7abe628c505c72f827dccfd47c7297f100;

struct MetavaultsRegistryState {
    mapping(address => MetavaultConfig) metavaults;
}

struct MarketInfosState {
    mapping(address => address) ptToMarket;
    mapping(address => address) ytToMarket;
    mapping(address => PoolInfos) poolToPoolInfos;
}

struct MetavaultsBridgeState {
    mapping(address => MetavaultBridgeConfig) metavaultBridgeConfigs;
}

function _getMarketInfosStorage() private pure returns (MarketInfosState storage $)
↪  {
    assembly {
        $.slot := MARKET_INFOS_LOCATION
```

```
    }
}

function _getMvConfigStorage() private pure returns (MetavaultsRegistryState
↪   storage $) {
    assembly {
        $.slot := MV_CONFIGS_LOCATION
    }
}

function _getMvBridgeStorage() private pure returns (MetavaultsBridgeState storage
↪   $) {
    assembly {
        $.slot := BRIDGE_CONFIGS_LOCATION
    }
}
```

Impact

Readability

Code Snippet

https://github.com/sherlock-audit/2025-12-spectra-metavault-update-dec-16th/pull/2/files#diff-d6dbd850708b46c46f64d8bc2d13e4fe2de6f7ac75f0981bf638274c7830a293R17-R58

Tool Used

Manual Review

Recommendation

Change the struct order

Discussion

petarcalic99

Agreed, will apply this

Issue L-6: address(0) check could be added to `MetavaultRegistry::registerMetavault()` [RESOLVED]

Source: https://github.com/sherlock-audit/2025-12-spectra-metavault-update-dec-16th/issues/12

Summary

It's technically possible to register a zero address metavault.

Vulnerability Detail

```
function registerMetavault(address metavault, address[] memory markets) external
↪  restricted {
    MetavaultConfig storage config = _getMvConfigStorage().metavaults[metavault];
    if (config.metavault != address(0)) revert
    ↪  MetavaultAlreadyRegistered(metavault);

    config.metavault = metavault;
    config.chainsCount = 1;

    // register the first markets to whitelist during the metavault registration
    for (uint256 i = 0; i < markets.length; i++) {
        config.markets.push(markets[i]);
        MarketConfig memory marketConfig = MarketConfig({isRegistered: true});
        config.marketConfigs[markets[i]] = marketConfig;
        _registerPoolInfos(markets[i]);
        emit MarketRegistered(metavault, markets[i]);
    }

    // Register current chain
    MetavaultChainConfig memory chainConfig = MetavaultChainConfig({
        chainId: block.chainid,
        remoteMetavaultAddress: metavault
    });
    config.chains.push(chainConfig);
    config.chainConfigs[block.chainid] = chainConfig;
    emit ChainRegistered(metavault, block.chainid, metavault);
    emit MetavaultRegistered(metavault);
}
```

Impact

It would be an admin mistake.

Code Snippet

https://github.com/sherlock-audit/2025-12-spectra-metavault-update-dec-16th/pull/2/files#diff-d6dbd850708b46c46f64d8bc2d13e4fe2de6f7ac75f0981bf638274c7830a293R83

Tool Used

Manual Review

Recommendation

Check if the metavault is not the zero address.

Discussion

petarcalic99

Yes we will ad a 0 address check.

Issue L-7: Some variable shadowing in MetavaultsRegistry [ACKNOWLEDGED]

Source: https://github.com/sherlock-audit/2025-12-spectra-metavault-update-dec-16th/issues/13

This issue has been acknowledged by the team but won't be fixed at this time.

Summary

`market` and `chainConfig` shadow the corresponding functions.

Vulnerability Detail

Market and chainConfig are used throughout the contract as memory variable, but there already exist functions with these names.

Impact

QA

Code Snippet

For example, market and market().

Tool Used

Manual Review

Recommendation

Consider changing the variable or function names.

Discussion

petarcalic99

Acknowledged, for now we will not change this to avoid changing our whole stack (tests, app, sdk etc..) as it doesn't cause any issues.

Issue L-8: `MetavaultsRegistry::registerMarket()/allowBridgePath()` duplicate `onlyRegistered` modifier [RESOLVED]

Source: https://github.com/sherlock-audit/2025-12-spectra-metavault-update-dec-16th/issues/14

Summary

Duplicate checks

Vulnerability Detail

`MetavaultsRegistry::onlyRegistered()` modifier is present on both the external functions and internal ones.

Impact

Gas usage.

Code Snippet

https://github.com/sherlock-audit/2025-12-spectra-metavault-update-dec-16th/pull/2/files#diff-d6dbd850708b46c46f64d8bc2d13e4fe2de6f7ac75f0981bf638274c7830a293R149

https://github.com/sherlock-audit/2025-12-spectra-metavault-update-dec-16th/pull/2/files#diff-d6dbd850708b46c46f64d8bc2d13e4fe2de6f7ac75f0981bf638274c7830a293R271

Tool Used

Manual Review

Recommendation

Keep the modifier only in the internal functions. Don't do this for the restricted modifier though, that one needs to be on the external function.

Issue L-9: Possible mistake in the unregister by index functions [ACKNOWLEDGED]

Source: https://github.com/sherlock-audit/2025-12-spectra-metavault-update-dec-16th/issues/15

This issue has been acknowledged by the team but won't be fixed at this time.

Summary

`MetavaultsRegistry::unregisterMarketByIndex()` don't check that the index matches the expected market, which may lead to issues if called twice in a row.

Vulnerability Detail

As `MetavaultsRegistry::unregisterMarketByIndex()` doesn't check the market, if 2 calls are made consecutively, they may arrive at the blockchain in different order, and remove the wrong markets (array popping changes order).

Impact

Code Snippet

https://github.com/sherlock-audit/2025-12-spectra-metavault-update-dec-16th/pull/2/files#diff-d6dbd850708b46c46f64d8bc2d13e4fe2de6f7ac75f0981bf638274c7830a293R187

Tool Used

Manual Review

Recommendation

Send the expected market as argument to verify.

Discussion

petarcalic99

Acknowledged, but we will keep this as for now to avoid refactoring the rest of the stack as this function is not really critical and there is the alternative version of the function that takes the market address parameter.

Issue L-10: `MetavaultsRegistry::registerChain()` allows chain id 0, which doesn't exist, but would be problematic [RESOLVED]

Source: https://github.com/sherlock-audit/2025-12-spectra-metavault-update-dec-16th/issues/16

Summary

`MetavaultsRegistry::registerChain()` if called with chain id 0, works, but `MetavaultsRegistry:unregisterChain()` would revert with chain not found.

Vulnerability Detail

`MetavaultsRegistry:unregisterChain()` reverting.

```
function unregisterChain(
    address metavault,
    uint256 chainId
) external restricted onlyRegistered(metavault) {
    MetavaultConfig storage config = _getMvConfigStorage().metavaults[metavault];
    if (config.chainConfigs[chainId].chainId == 0)
        revert ChainNotRegistered(metavault, chainId);
```

Impact

Chain id == 0, doesn't exist so it would be admin mistake.

Code Snippet

https://github.com/sherlock-audit/2025-12-spectra-metavault-update-dec-16th/pull/2/files#diff-d6dbd850708b46c46f64d8bc2d13e4fe2de6f7ac75f0981bf638274c7830a293R202

Tool Used

Manual Review

Recommendation

Don't allow chain id == 0 in `MetavaultsRegistry::registerChain()`.

Discussion

petarcalic99

Yes, we will ad a 0 chain id check

Issue L-11: `MetavaultsRegistry::removeBridgePathByIndex()` is missing [RESOLVED]

Source: https://github.com/sherlock-audit/2025-12-spectra-metavault-update-dec-16th/issues/17

Summary

`MetavaultsRegistry::removeBridgePathByIndex()` doesn't exist, which could cause DoS.

Vulnerability Detail

All other functions to remove data have a ”by index” counterpart, but `MetavaultsRegistry::removeBridgePath()` doesn't.

Impact

Possible DoS.

Code Snippet

https://github.com/sherlock-audit/2025-12-spectra-metavault-update-dec-16th/pull/2/files#diff-d6dbd850708b46c46f64d8bc2d13e4fe2de6f7ac75f0981bf638274c7830a293R296

Tool Used

Manual Review

Recommendation

Add the ”by index” equivalent.

Discussion

petarcalic99

Agreed, the internal function already exists so we will implement it.

Disclaimers

Sherlock does not provide guarantees nor warranties relating to the security of the project.

Usage of all smart contract software is at the respective users' sole risk and is the users' responsibility.