SHELLBOXES

Velvet Capital V2 - Arbitrum Version

Smart Contract Security Audit

Prepared by ShellBoxes

Nov 23rd, 2023 - Nov 29th, 2023

Shellboxes.com

contact@shellboxes.com

Document Properties

Client	Velvet Capital
Version	1.0
Classification	Public

Scope

Repository	Commit Hash
https://github.com/Velvet-Capital/protocol-v2-public	e14c7cd02f712ff0e4fae050d89983f04e1d9bd7

Re-Audit

Repository	Commit Hash
https://github.com/Velvet-Capital/protocol-v2-public	385d5769974b546f051afabb568a01bb510ce533

Contacts

COMPANY	EMAIL
ShellBoxes	contact@shellboxes.com

Contents

1 Introduction

Velvet Capital engaged ShellBoxes to conduct a security assessment on the Velvet Capital V2 - Arbitrum Version beginning on Nov 23rd, 2023 and ending Nov 29th, 2023. In this report, we detail our methodical approach to evaluate potential security issues associated with the implementation of smart contracts, by exposing possible semantic discrepancies between the smart contract code and design document, and by recommending additional ideas to optimize the existing code. Our findings indicate that the current version of smart contracts can still be enhanced further due to the presence of many security and performance concerns.

This document summarizes the findings of our audit.

1.1 About Velvet Capital

Velvet Capital is a DeFi protocol that helps people & institutions create tokenized index funds, portfolios & other financial products with additional yield. The protocol provides all the necessary infrastructure for financial product development being integrated with AMMs, Lending protocols and other DeFi primitives to give users a diverse asset management toolkit.

Issuer	Velvet Capital
Website	`https://www.velvet.capital`
Type	Solidity Smart Contracts
Documentation	Arbitrum New Handlers
Audit Method	Whitebox

1.2 Approach & Methodology

ShellBoxes used a combination of manual and automated security testing to achieve a balance between efficiency, timeliness, practicability, and correctness within the audit's scope. While manual testing is advised for identifying problems in logic, procedure, and implementation, automated testing techniques help to expand the coverage of smart contracts and can quickly detect code that does not comply with security best practices.

1.2.1 Risk Methodology

Vulnerabilities or bugs identified by ShellBoxes are ranked using a risk assessment technique that considers both the LIKELIHOOD and IMPACT of a security incident. This framework is effective at conveying the features and consequences of technological vulnerabilities.

Its quantitative paradigm enables repeatable and precise measurement, while also revealing the underlying susceptibility characteristics that were used to calculate the Risk scores. A risk level will be assigned to each vulnerability on a scale of 5 to 1, with 5 indicating the greatest possibility or impact.

— Likelihood quantifies the probability of a certain vulnerability being discovered and exploited in the untamed.

— Impact quantifies the technical and economic costs of a successful attack.

— Severity indicates the risk's overall criticality.

Probability and impact are classified into three categories: H, M, and L, which correspond to high, medium, and low, respectively. Severity is determined by probability and impact and is categorized into four levels, namely Critical, High, Medium, and Low.

Impact			
High	Critical	High	Medium
Medium	High	Medium	Low
Low	Medium	Low	Low
	High	Medium	Low
		Likelihood	

2 Findings Overview

2.1 Summary

The following is a synopsis of our conclusions from our analysis of the Velvet Capital V2 - Arbitrum Version implementation. During the first part of our audit, we examine the smart contract source code and run the codebase via a static code analyzer. The objective here is to find known coding problems statically and then manually check (reject or confirm) issues highlighted by the tool. Additionally, we check business logics, system processes, and DeFi-related components manually to identify potential hazards and/or defects.

2.2 Key Findings

In general, these smart contracts are well-designed and constructed, but their implementation might be improved by addressing the discovered flaws, which include , 1 medium-severity, 1 low-severity, 1 informational-severity vulnerabilities.

Vulnerabilities	Severity	Status
SHB.1. Lack of Sequencer Availability Check in Arbitrum	MEDIUM	Fixed
SHB.2. Missing Address Verification	LOW	Fixed
SHB.3. Inefficient ExternalSwapHandler Contracts Implementation	INFORMATIONAL	Acknowledged

3 Finding Details

SHB.1 Lack of Sequencer Availability Check in Arbitrum

- Severity : MEDIUM
- Status : Fixed

- Likelihood : 1
- Impact : 3

Description:

The protocol does not include a mechanism to verify the status of the Arbitrum sequencer. In optimistic rollup protocols, the sequencer is crucial for batching L2 transactions. Its unavailability can disrupt the access to read/write APIs, significantly impacting user interactions with the L2 network. You can review Chainlink docs on L2 Sequencer Uptime Feeds for more details on this.

Files Affected:

SHB.1.1: PriceOracle.sol

```
function latestRoundData(address base, address quote) internal view returns (int256) {
  (
    ,
    /*uint80 roundID*/
    int256 price /*uint startedAt*/ /*uint timeStamp*/ /*uint80 answeredInRound*/,
    ,
    uint256 updatedAt,

  ) = aggregatorAddresses[base].aggregatorInterfaces[quote].latestRoundData();

  if (updatedAt + oracleExpirationThreshold < block.timestamp) {
    revert ErrorLibrary.PriceOracleExpired();
  }

  if (price == 0) {
    revert ErrorLibrary.PriceOracleInvalid();
  }

  return price;
}
```

Recommendation:

Implement a sequencer status check using an oracle data feed. This feed would provide real-time information about the sequencer's availability, allowing the protocol to respond accordingly.

Updates

The team resolved the issue by creating a new oracle called PriceOracleL2, this oracle has the necessary checks for Arbitrum sequencer's state before using the price feed.

SHB.1.2: PriceOracle.sol

```
  function latestRoundData(address base, address quote) internal view
    ↪ returns (int256) {
    (
      ,
      /*uint80 roundID*/ int256 answer,
      uint256 startedAt /*uint256 updatedAt*/ /*uint80 answeredInRound*/
        ↪ ,
      ,

    ) = sequencerUptimeFeed.latestRoundData();

    //Checking whether sequencer is up or not
    bool isSequencerUp = answer == 0;
    if (!isSequencerUp) {
      revert ErrorLibrary.SequencerIsDown();
    }

    //Checking Whether Sequncer Threshold Is Crossed After Uptime
    if (block.timestamp - startedAt <= sequencerThreshold) {
      revert ErrorLibrary.SequencerThresholdNotCrossed();
    }

    (
      ,
      /*uint80 roundID*/
      int256 price /*uint startedAt*/ /*uint timeStamp*/ /*uint80
        ↪ answeredInRound*/,
      ,
      uint256 updatedAt,

    ) = aggregatorAddresses[base].aggregatorInterfaces[quote].
       ↪ latestRoundData();

    if (updatedAt + oracleExpirationThreshold < block.timestamp) {
      revert ErrorLibrary.PriceOracleExpired();
    }

    if (price == 0) {
      revert ErrorLibrary.PriceOracleInvalid();
    }

    return price;
  }
```

SHB.2 Missing Address Verification

- Severity : LOW
- Status : Fixed
- Likelihood : 1
- Impact : 2

Description:

Certain functions within the smart contracts lack a crucial safety check in the address parameter, specifically the absence of a zero-address test. This omission poses a potential risk, as the contract's functionality may become inaccessible.

- _rewardContract in the AaveV3Handler and CompoundV3Handlercontracts constructor.
- _protocol_Handler in the BeefyBridgeHandler contract constructor.
- _moo_eth in the BeefyYieldHandler contract constructor.
- _swapTarget and _oracle in the BebopHandler and KyberSwapHandler contracts's init function.
- comptroller in the VenusHandler contract constructor.
- _wombat_optimized_proxy and _wombat_router in the WombatHandler contract constructor.

Files Affected:

SHB.2.1: AaveV3Handler.sol

```
  constructor(address _priceOracle, address aave_pool, address
      ↪ token_gateway, address _rewardContract) {
    if (_priceOracle == address(0)  aave_pool == address(0)
        ↪ token_gateway == address(0))
      revert ErrorLibrary.InvalidAddress();
    _oracle = IPriceOracle(_priceOracle);
  AAVE_POOL_V3 = aave_pool;
  WRAPPED_TOKEN_GATEWAY = token_gateway;
  REWARD_CONTRACT = _rewardContract;
```

SHB.2.2: CompoundV3Handler.sol

```
constructor(address _priceOracle, address _rewardContract) {
  if (_priceOracle == address(0)) revert ErrorLibrary.InvalidAddress()
     ↪ ;
  _oracle = IPriceOracle(_priceOracle);
  _cometReward = _rewardContract;
```

SHB.2.3: BeefyBridgeHandler.sol

```
constructor(address _priceOracle,address _moo_eth,address
    ↪ _protocol_Handler) {
  if (_priceOracle == address(0)   _moo_eth == address(0)) revert
     ↪ ErrorLibrary.InvalidAddress();
  _oracle = IPriceOracle(_priceOracle);
  MOO_ETH = _moo_eth;
  WETH = _oracle.WETH();
  Protocol_Handler = _protocol_Handler;
```

SHB.2.4: BeefyYieldHandler.sol

```
constructor(address _priceOracle,address _moo_eth) {
  if(_priceOracle == address(0)){
    revert ErrorLibrary.InvalidAddress();
  }
  _oracle = IPriceOracle(_priceOracle);
  WETH = _oracle.WETH();
  MOO_ETH = _moo_eth;
```

SHB.2.5: BebopHandler.sol

```
function init(address _swapTarget, address _oracle) external
    ↪ initializer {
  swapTarget = _swapTarget;
```

```
    oracle = IPriceOracle(_oracle);
```

SHB.2.6: KyberSwapHandler.sol

```
  function init(address _swapTarget, address _oracle) external
      ↪ initializer {
    swapTarget = _swapTarget;
    oracle = IPriceOracle(_oracle);
```

SHB.2.7: VenusHandler.sol

```
  constructor(address _priceOracle, address comptroller) {
    if(_priceOracle == address(0)){
      revert ErrorLibrary.InvalidAddress();
    }
    _oracle = IPriceOracle(_priceOracle);
    COMPTROLLER = comptroller;
```

SHB.2.8: WombatHandler.sol

```
  constructor(address _priceOracle, address _wombat_optimized_proxy,
      ↪ address _wombat_router) {
    if(_priceOracle == address(0)){
      revert ErrorLibrary.InvalidAddress();
    }
    WOMBAT_OPTIMIZED_PROXY = _wombat_optimized_proxy;
    MasterWombat = IWombat(_wombat_optimized_proxy);

    WOMBAT_ROUTER = _wombat_router;
```

Recommendation:

We recommend that you make sure the addresses provided in the arguments are different from the address(0). Additionally, enhance the security posture by incorporating a contract address check, utilizing the isContract modifier from the OpenZeppelin library. Import the Address contract from OpenZeppelin to facilitate robust contract address validation.

Updates

The team resolved the issue by adding the necessary zero-address checks.

SHB.3 Inefficient ExternalSwapHandler Contracts Implementation

- Severity : INFORMATIONAL
- Status : Acknowledged
- Likelihood : 1
- Impact : 0

Description:

The external swap handler contracts includes an unnecessary receive function, leading to locked funds. Additionally, the contracts declares a swap function as payable without utilizing the msg.value parameter in its logic.

Files Affected:

SHB.3.1: BebopHandler.sol

```
receive() external payable {}
```

SHB.3.2: KyberSwapHandler.sol

```
receive() external payable {}
```

SHB.3.3: OneInchHandler.sol

```
receive() external payable {}
```

SHB.3.4: ParaswapHandler.sol

```
receive() external payable {}
```

SHB.3.5: ZeroExHandler.sol

```
receive() external payable {}
```

Recommendation:

Consider removing the unnecessary receive function to prevent fund lockups. Additionally, eliminate the payable keyword from the swap functions.

Updates

The team acknowledged the issue, as they would like to keep the codebase consistent with the BNB version.

4 Best Practices

BP.1 Enhancing Variable Naming for Clarity

Description:

To improve code readability, consider renaming the variable t in the getTokenBalance and getUnderlyingBalance functions to a more descriptive name. Choosing a name that reflects the purpose or nature of the variable enhances code clarity and aids in understanding the function's functionality.

Files Affected:

BP.1.1: AaveV3Handler.sol

```
function getTokenBalance(address _tokenHolder, address t) public view
    ↪ override returns (uint256 tokenBalance) {
```

BP.1.2: AaveV3Handler.sol

```
function getUnderlyingBalance(address _tokenHolder, address t) public
    ↪ view override returns (uint256[] memory) {
```

Status - Acknowledged

BP.2 Eliminating Duplicate Address Verification Checks

Description:

To enhance the clarity and efficiency of the getUnderlying function in the BeefyBridgeHandler contract, eliminate the redundant address verification check for _mooAsset. The duplicate check is unnecessary and can be safely removed without compromising the function's integrity.

Files Affected:

BP.2.1: BeefyBridgeHandler.sol

```
function getUnderlying(address _mooAsset) public view override returns
    ↪  (address[] memory) {
  if (address(_mooAsset) == address(0)) revert ErrorLibrary.
      ↪ InvalidAddress();
  if (_mooAsset == address(0)) {
    revert ErrorLibrary.InvalidAddress();
  }
```

Status - Acknowledged

BP.3 Optimizing Variable Usage and Enhancing Code Readability in the swap Function

Description:

To improve the efficiency and readability of the swap function in the BebopHandler contract, consider directly utilizing the buyTokenBalance variable within the safeTransfer function. This eliminates the need for redundant calls to IERC20Upgradeable(buyTokenAddress).balanceOf(address(this)). The suggested modification streamlines the code, reduces gas consumption, and enhances overall code clarity.

Files Affected:

BP.3.1: BebopHandler.sol

```
uint buyTokenBalance = IERC20Upgradeable(buyTokenAddress).balanceOf(
    ↪ address(this));
tokensSwapped = buyTokenBalance - tokensBefore;
if (tokensSwapped == 0) {
  revert ErrorLibrary.ZeroTokensSwapped();
```

```
    }
    uint priceSellToken = oracle.getPriceTokenUSD18Decimals(
        ↪ sellTokenAddress, sellAmount);
    uint priceBuyToken = oracle.getPriceTokenUSD18Decimals(
        ↪ buyTokenAddress, buyTokenBalance);

    validateSwap(priceSellToken, priceBuyToken);
    TransferHelper.safeTransfer(buyTokenAddress, _to, IERC20Upgradeable(
        ↪ buyTokenAddress).balanceOf(address(this)));
```

Status - Acknowledged

BP.4 Remove Dead Code for Enhanced Code Maintainability

Description:

To improve code maintainability in the HopHandler contract, it is recommended to remove dead code within the getToken function. Dead code, can lead to confusion and should be eliminated for a cleaner and more understandable codebase.

Files Affected:

BP.4.1: HopHandler.sol

```
// function getToken() public returns()
```

Status - Acknowledged

BP.5 Simplify The removePidMap Function Parameters

Description:

To improve code readability and simplify the logic in the removePidMap function of the SushiSwapLPHandler contract, consider modifying the function to accept only _lpTokens as a parameter. Additionally, ensure that the pidMap function includes a non-zero value assignment. By doing this, there is no need to include an additional check for _pid in the removePidMap function. The modified function exclusively verifies the existence of values in the pid mapping and removes the corresponding entries.

Files Affected:

BP.5.1: SushiSwapLPHandler.sol

```
function removePidMap(address[] memory _lpTokens, uint256[] memory
    ↪ _pid) external onlyOwner {
  if (_lpTokens.length != _pid.length) {
    revert ErrorLibrary.InvalidLength();
  }
  for (uint256 i = 0; i < _lpTokens.length; i++) {
    if (pid[_lpTokens[i]] != _pid[i]) {
      revert ErrorLibrary.InvalidPID();
    }
    delete (pid[_lpTokens[i]]);
  }
}
```

Status - Acknowledged

BP.6 Adding the immutable Keyword to Constant Variables Assigned in the Constructor

Description:

To enhance gas efficiency and emphasize the immutability of constant variables assigned in the constructor, it is recommended to add the immutable keyword to the WOMBAT_OPTIMIZED_PROXY and WOMBAT_ROUTER addresses in the WombatHandler contract. The use of the immutable keyword signals to the compiler that these values are known at compile-time and will not change throughout the contract's lifetime.

Files Affected:

BP.6.1: WombatHandler.sol

```
address public WOMBAT_OPTIMIZED_PROXY;
IWombat public MasterWombat;

address public WOMBAT_ROUTER;
```

Status - Acknowledged

5 Tests

Results:

```
→ Tests running for Handler: Aave
✓ should lend tokens (23730ms)
✓ return values of deposit should be greater than 0 (232ms)
✓ should redeem tokens (3635ms)
✓ gets underlying asset of the token
✓ should get token balance of the token holder
✓ should get the token price in USD (57ms)
→ Tests running for Handler: Aav
✓ should lend tokens (18095ms)
✓ return values of deposit should be greater than 0 (225ms)
✓ should redeem tokens (1715ms)
✓ gets underlying asset of the token
✓ should get token balance of the token holder
✓ should get the token price in USD (50ms)
→ Tests running for Handler: ApeSwap-lp
✓ should lend tokens (17730ms)
✓ return values of deposit should be greater than 0 (431ms)
✓ should redeem tokens (3524ms)
✓ gets underlying asset of the token
✓ should get token balance of the token holder
✓ should get the token price in USD (84ms)
→ Tests running for Handler: ApeSwap-lp
✓ should lend tokens (12747ms)
✓ return values of deposit should be greater than 0 (518ms)
✓ should redeem tokens (1597ms)
✓ gets underlying asset of the token
✓ should get token balance of the token holder
✓ should get the token price in USD (96ms)
→ Tests running for Handler: Hop
✓ should lend tokens (11738ms)
✓ return values of deposit should be greater than 0 (274ms)
✓ should redeem tokens (952ms)
✓ gets underlying asset of the token
✓ should get token balance of the token holder
✓ should get the token price in USD (125ms)
→ Tests running for Handler: Hop
✓ should lend tokens (15436ms)
✓ return values of deposit should be greater than 0 (184ms)
✓ should redeem tokens (1011ms)
✓ gets underlying asset of the token
✓ should get token balance of the token holder
✓ should get the token price in USD (2427ms)
→ Tests for Beefy Handler
✓ should lend tokens mooHopEth (46777ms)
✓ should lend tokens mooHopUsdc (44631ms)
✓ should redeem tokens (2492ms)
✓ gets underlying asset of the token
✓ should get token balance of the token holder
✓ should get the token price in USD (159ms)
→ Tests for BeefyLP Handler
✓ should lend tokens mooSushiWethUsdc (37658ms)
✓ should redeem tokens (2260ms)
✓ gets underlying asset of the token
✓ should get token balance of the token holder
✓ should get the token price in USD (532ms)
→ Tests for Compund Handler
✓ should lend tokens (9129ms)
✓ should redeem tokens (850ms)
```

✓ gets underlying asset of the token

✓ should get token balance of the token holder

✓ should get the token price in USD

→ Tests for Mock Fee

✓ should revert back if the custodial is true and no address is passed in _owner (43ms)

✓ should revert back if the _custodial is true and threshold is more than owner length

✓ Initialize 1st IndexFund Tokens (85ms)

✓ Calculate fees should return fee values

✓ Invest 1 WETH into Top10 fund (6794ms)

✓ Should charge fees for index 1 (258ms)

✓ Should charge fees for index 1 (244ms)

→ Tests for IndexFactory contract

✓ should revert back if the custodial is true and no address is passed in _owner

✓ should revert back if the _custodial is true and threshold is more than owner length

✓ should revert is zeroAddress is passed (1062ms)

✓ should revert if token is not enabled

✓ asset manager should create a private transferable fund and make it non-transferable (3144ms)

```
✓ asset manager should be able to make the previous private fund transferable to whitelisted addresses (63ms)
✓ asset manager should be able to convert the previous transferable private fund to public (61ms)
✓ asset manager should be able to make the previous public fund non-transferable (39ms)
✓ asset manager should not be able to make the previous public fund transferable to only whitelisted addresses (42ms)
✓ asset manager should be able to make the previous public fund transferable (41ms)
✓ should check Index token name and symbol
✓ should check if module owner of all fund is exchange contract
✓ initialize should revert if total Weights not equal 10,000 (56ms)
✓ initialize should revert if tokens and denorms length is not equal
✓ initialize should revert if token not whitelisted
✓ Initialize 1st IndexFund Tokens (45ms)
✓ Initialize 2nd IndexFund Tokens (43ms)
✓ Initialize 3rd IndexFund Tokens (47ms)
✓ Initialize 4th IndexFund Tokens (44ms)
✓ Initialize 5th IndexFund Tokens (44ms)
✓ Initialize 6th IndexFund Tokens (43ms)
```

✓ Owner of vault for 1st fund should be exchangeHandler address

✓ Owner of vault for 2nd fund should be deployer’s addressess

✓ Owner of vault for 3rd fund should be exchangeHandler address

✓ Owner of vault for 4th fund should be exchangeHandler address

✓ Calculate fees should return fee values

✓ expect owner to be IndexFactory (47ms)

✓ Invest 0.1WETH into Top10 fund should fail for slippage greater than 10

✓ Invest 0.1WETH into Top10 fund (481ms)

✓ Invest 0.1WETH into Top10 fund 5th (4549ms)

✓ Invest 1WETH into Top10 fund 6th (28013ms)

✓ Invest 2WETH into Top10 2nd index fund (382ms)

✓ Invest 0.1WETH into Top10 3rd index fund (4813ms)

✓ Invest 0.2WETH into Top10 4th index fund (8403ms)

✓ Invest 2WETH into Top10 4th index fund should revert if WETH value is greater than 0 and investment token is not WETH (427ms)

✓ Invest 0.2WETH into Top10 4th index fund on behalf of addr3 should fail if user addr3 is not whitelisted

✓ Add addr3 whitelisted user

✓ Invest 2WETH into Top10 4th index fund (4813ms)

✓ Invest 0.1WETH into Top10 fund (373ms)

✓ Invest 0.1WETH into Top10 fund (371ms)

✓ Invest 0.1WETH into Top10 fund (377ms)

✓ Invest 0.1WETH into Top10 fund (419ms)

✓ Invest 0.1WETH into Top10 fund (382ms)

✓ Add addr1 whitelisted user (47ms)

✓ non owner should not be able to add whitelist manager admin

✓ owner should be able to add asset whitelist manager admin

✓ owner should not be able to add index manager

✓ owner should not be able to add rebalancing manager

✓ non whitelist manager admin should not be able to add asset manager

✓ new whitelist manager admin should be able to add whitelist manager

✓ owner should be able to add whitelist manager

✓ non whitelist manager should not be able to update merkle root

✓ Whitelist manager should be able to update merkle root (46ms)

✓ Whitelist manager should be able to add and remove a whitelisted user (40ms)

✓ non whitelist manager admin should not be able to revoke whitelist manager

✓ whitelist manager admin should be able to revoke whitelist manager

✓ Whitelist manager should not be able to add user to whitelist after his role was revoked

✓ New (addr1) whitelisted user invest 0.2 WETH into Top10 2nd index fund (1409ms)

✓ New (addr2) whitelisted user invest 2WETH into Top10 2nd index fund (401ms)

✓ Non whitelisted user invest 2WETH into Top10 2nd index fund should fail

✓ Should charge fees for index 1 (212ms)

✓ Should charge fees for index 2 (156ms)

✓ Management fees for index 3 should be 0 (144ms)

✓ Invest 0.00001 WETH into Top10 fund should fail (53ms)

✓ asset manager should be able to add token which is approved in registry for all the indexes (113ms)

✓ Invest 2WETH into Top10 fund (372ms)

✓ Invest 1WETH into Top10 2nd Index fund (396ms)

✓ Invest 1WETH into Top10 fund (366ms)

✓ Invest 1WETH into Top10 2nd Index fund (412ms)

✓ Investment should fail when contract is paused

✓ update Weights should revert if total Weights not equal 10,000

✓ Update Weights and Rebalance should revert if one of the weight is zero

✓ should Update Weights and Rebalance (2560ms)

✓ should Update Weights and Rebalance for 2nd Index Fund (1020ms)

✓ should Update Weights and Rebalance (4120ms)

```
✓ updateTokens should revert if total Weights not equal 10,000 (279ms)
✓ updateTokens should revert if token is not whitelisted (2663ms)
✓ updateTokens should revert if token is not enabled
✓ updateTokens should revert if protocol is paused
✓ updateTokens should revert if swapHandler is not enabled (45ms)
✓ Non Rebalancing access address calling update function
✓ update tokens should revert is any two tokens are same (266ms)
✓ should update tokens (4904ms)
✓ print values (69ms)
✓ should update tokens (2190ms)
✓ withdrawal should revert when contract is paused
✓ should unpause
✓ should pause
✓ should revert unpause
✓ should unpause
✓ should update tokens for 2nd Index (2890ms)
✓ when withdraw fund more then balance (55ms)
✓ should fail withdraw when balance falls below min investment amount (223ms)
✓ should fail withdraw when balance falls below min investment amount (190ms)
✓ should withdraw fund and burn index token successfully (1234ms)

✓ should withdraw fund and burn index token successfully (1597ms)

✓ should withdraw fund and burn index token successfully for account
  that has been removed from whitelist (524ms)

✓ Invest 0.1WETH into Top10 2nd Index fund (463ms)

✓ transfer idx for a non transferable portfolio should fail (42ms)

✓ transfer idx from owner to non whitelisted account should fail

✓ transfer idx from owner to a whitelisted account

✓ transfer idx from owner to another account (Index 3)

✓ transfer idx from owner to another account (Index 4)

✓ new owner of idx withdraws funds from Index 3 (1804ms)

✓ Invest 1WETH into Top10 fund after last withdrawal (335ms)

✓ withdraw check values (353ms)

✓ should withdraw fund and burn index token successfully for 2nd Index
  (537ms)

✓ Invest 2WETH into Top10 fund (491ms)

✓ Invest 0.1WETH into Top10 fund (457ms)

✓ Invest 0.1WETH into Top10 2nd Index fund (476ms)

✓ Invest 0.1WETH into Top10 2nd Index fund (476ms)

✓ should withdraw tokens directly instead of WETH (1370ms)

✓ should withdraw tokens directly instead of WETH for 2nd Index (328ms)
```

✓ non owner should not be able to add asset manager admin

✓ owner should be able to add asset manager admin

✓ non asset manager admin should not be able to add asset manager

✓ new asset manager admin should be able to add asset manager

✓ owner should be able to add asset manager

✓ non-owner should be able to pause protocol

✓ should not upgrade Proxy Exchnage To New Contract for 1st Index

✓ should protocol pause

✓ should upgrade Proxy Exchnage To New Contract for 1st Index and 2nd Index

✓ should not upgrade if msg.sender is not owner

✓ non owner of indexFactory should not be able to upgrade Exchange

✓ should upgrade Proxy IndexSwap To New Contract for 1st Index

✓ should upgrade Proxy OffChainIndexSwap To New Contract for 1st Index (45ms)

✓ should unpause protocol

✓ Invest 2WETH into Top10 1st index fund after upgrade (424ms)

✓ Invest 2WETH into Top10 1st index fund after upgrade (420ms)

✓ should pause protocol

✓ should upgrade Proxy IndexSwap To New Contract for 2nd Index

✓ should unpause protocol

✓ Invest 2WETH into Top10 2nd index fund after upgrade (556ms)

✓ Upgrade TokenRegistry (359ms)

✓ Upgrade IndexFactory, and not able to create Index (85ms)

✓ should unpause index creation and creat index (8408ms)

✓ should set new cool down period

× Invest 1WETH into Top10 2nd index fund after upgrade and should no revert

× should withdraw fund and burn index token successfully should fail

× transfer tokens should fail, if cooldownperiod is not passed

× should transfer token and withdraw fund and burn index token successfully

✓ should fail to create an index with management fee greater than max fee (349ms)

✓ should fail to create an index with entry fee greater than max fee (354ms)

✓ should fail to create an index with exit fee greater than max fee (392ms)

✓ should fail to create an index with management fee greater than max fee (374ms)

✓ Non asset manager should not be able to propose new management fee

✓ Asset manager should propose new management fee

✓ Asset manager should not be able to update management fee before 28 days passed

✓ Non asset manager should not be able to delete proposed new management fee (39ms)

✓ Asset manager should be able to delete proposed new management fee

✓ Non asset manager should not be able to update management fee

✓ asset manager should not be able to update management without proposing new fees

✓ Non asset manager should not be able to propose new performance fee

✓ Asset manager should propose new performance fee

✓ Asset manager should be able to update performance fee before 28 days passed

✓ Non asset manager should not be able to delete proposed new performance fee

✓ Asset manager should be able to delete proposed new performance fee

✓ Non asset manager should not be able to update performance fee

✓ asset manager should not be able to update performance without proposing new fees

✓ Non asset manager should not be able to propose new entry fee

✓ Asset manager should propose new entry and exit fee (38ms)

✓ Asset manager should be able to update entry and exit fee before 28 days passed

✓ Non asset manager should not be able to delete proposed new entry and exit fee

✓ Asset manager should be able to delete proposed new entry and exit fee

✓ Non asset manager should not be able to update entry and exit fee

✓ asset manager should not be able to update entry and exit fee without proposing new fees

✓ Non asset manager should not be able to update the asset manager treasury

✓ Asset manager should not be able to update the asset manager treasury

✓ Non asset manager should not be able to update the velvet treasury

✓ Asset manager should be able to update the velvet treasury

✓ Non owner should not be able to update protocol slippage

✓ Owner should not be able to update to a slippage more than 10

✓ should claim tokens (3552ms)

✓ should swap reward token (4475ms)

→ Tests for MixedIndex

✓ should check Index token name and symbol

✓ initialize should revert if total Weights not equal 10,000 (56ms)

✓ Initialize should fail if the number of tokens exceed the max limit set during deployment (current = 15)

✓ should retrieve the current max asset limit from the TokenRegistry

✓ should update the max asset limit to 10 in the TokenRegistry

✓ should retrieve the current max asset limit from the TokenRegistry

✓ Initialize should fail if the number of tokens exceed the max limit set by the Registry (current = 10)

✓ Initialize IndexFund Tokens (62ms)

✓ should add pid

✓ should remove pid

✓ asset manager should not be able to add token which is not approved in registry

✓ Invest 0.16 WETH should not revert , if investing token is not initialized (52725ms)

✓ Invest 10 USDT should revert , if investing token is not initialized (433ms)

✓ asset manager should be able to add token which is approved in registry (40ms)

✓ Invest 0.1 WETH into Top10 fund (1636ms)

✓ Invest 10 USDT into Top10 fund (4363ms)

✓ Invest 0.00001 WETH into Top10 fund should fail (124ms)

✓ Should change expectedRangeDecimal

✓ Invest 2 WETH into Top10 fund (1298ms)

✓ should return false if both of the token in pool is not WETH

✓ Invest 1 WETH into Top10 fund (1316ms)

```
✓ Investment should fail when contract is paused
✓ update Weights should revert if total Weights not equal 10,000 (51ms)
✓ should Update Weights and Rebalance (4365ms)
✓ updateTokens should revert if total Weights not equal 10,000 (424ms)
✓ owner should be able to add asset manager
✓ non owner should not be able to add asset manager (48ms)
✓ new asset manager should update tokens (63451ms)
✓ withdrawal should revert when contract is paused
✓ should unpause
✓ should pause
✓ should revert unpause
✓ should unpause
✓ when withdraw fund more then balance (54ms)
✓ should fail withdraw when balance falls below min investment amount
  (435ms)
✓ should fail withdraw when balance falls below min investment amount
  (multi asset) (526ms)
✓ should withdraw fund and burn index token successfully (18350ms)
✓ Invest 0.1WETH into Top10 fund (2337ms)
✓ Invest 0.1WETH into Top10 fund (1347ms)
✗ Invest 1WETH into Top10 fund
```

✓ Invest 1 WETH into Top10 fund (1448ms)

✓ should withdraw fund in ETH and burn index token successfully (1569ms)

✓ Invest 0.1 WETH into Top10 fund (1351ms)

✓ Invest 0.1 WETH into Top10 fund (1311ms)

✓ should withdraw tokens directly instead of WETH (3765ms)

→ Tests for MixedIndex

✓ should check Index token name and symbol

✓ initialize should revert if tokens length does not match denorms length

✓ initialize should revert if a token address is null (43ms)

✓ initialize should revert if a non-approved token is being used for init (53ms)

✓ initialize should revert if total Weights not equal 10,000 (97ms)

✓ Initialize IndexFund Tokens (94ms)

✓ Initialize 2nd IndexFund Tokens (208ms)

✓ should confirm that the correct tokens are initialised for the 1st index (41ms)

✓ should update a price Oracle feed

✓ should not be able to add pid if arrray lengths don't match

✓ should not be able to delete pid if array lengths don't match

✓ should add pid

✓ should not be able to delete pid if the entry does not match (49ms)

✓ should delete pid (49ms)

✓ should fetch the router address of the pancake LP handler

✓ should check if a token is enabled or not in the registry

✓ should disable a token in the registry

✓ should reiterate the WETH address of the token registry

✓ should not be able to enable a zero address permitted token in TokenRegistry

✓ should not be able to enable if empty array is passed to TokenRegistry

✓ should not be able to enable a token which is already enabled

✓ should not be able to enable token in registry if the oracle array length does not match the length of other arrays

✓ should not be able to enable token in registry if the token array length does not match the length of other arrays

✓ should not be able to enable token in registry if the handler array length does not match the length of other arrays

✓ should not be able to enable token in registry if the reward token array length does not match the length of other arrays (40ms)

✓ should not be able to enable token in registry if the primary token array length does not match the length of other arrays

✓ disable token in registry should fail if zero address is passed

✓ disable token in registry should fail if token is not enabled at all

✓ disable token in registry should fail if empty array is passed

✓ Non owner should not be able to disable a permitted token in TokenRegistry

✓ should successfully disable a permitted token in TokenRegistry

✓ should update an enabled token's data in the TokenRegistry (1971ms)

✓ should revert if wrong LP slippage is assigned to a LP handler (450ms)

✓ should revert if a zero address is passed to ApeSwapLP Handler's constructor (423ms)

✓ should revert if a zero address is passed to Compund Handler's constructor

✓ should not be able to enable swap handlers in registry if an empty array is passed

✓ should not be able to enable a handler in the registry that has already been enabled

✓ Non-primary tokens should not get enabled on the registry level

✓ asset manager should not be able to add token which is not approved in registry (46ms)

✓ asset manager should not be able to add a zero address as permitted token

✓ asset manager should not be able to delete a non-permitted token (54ms)

✓ asset manager should not be able to delete permitted tokens if an empty array is passed

✓ isTokenPermitted should not return output for zero address (50ms)

✓ Invest 0.1 WETH should not revert, if investing token is not initialized + tranferFrom should not work (9785ms)

✓ Invest 0.1 WETH in 2nd index (53296ms)

✓ Invest 1 WETH in 2nd index (2791ms)

✓ Invest 10 USDT should revert if investing token is not initialized (447ms)

✓ asset manager should be able to permit token which is approved in registry (48ms)

✓ should not be able to get underlying token of a zero address Wombat lp token

✓ should not be able to get token balance of a zero address Wombat lp token

✓ should not be able to get token balance of a zero address Wombat lp token holder

✓ should not be able to get underlying balance of a zero address Wombat lp token

✓ should not be able to get underlying balance of a zero address Wombat lp token holder

✓ should add reward token to registry and verify it (51ms)

✓ should remove reward token from registry and verify it

✓ should add reward token to registry and verify it

✓ should revert when add reward token to registry sending 0 address token address

✓ should revert when add reward token to registry sending 0 address handler address

✓ Invest 10 USDCe into Top10 fund (648ms)

✓ Invest 0.00001 WETH into Top10 fund should fail (77ms)

✓ Invest 10 WETH into Top10 fund (548ms)

✗ Invest 10 WETH into Top10 fund

✓ Investment should fail when contract is paused (63ms)

✓ should be able to claim tokens for portfolio tokens (120ms)

✓ update Weights should revert if total Weights not equal 10,000 (42ms)

✓ update weights should revert if weights and slippage array length don't match

✓ update weights should revert if slippage array length don't match the token count

✓ update weights should revert if swap handler is not enabled

✓ update Weights should revert if total Weights not equal 10,000 (63ms)

✓ update weights should revert if weights and slippage array length don't match

✓ update weights should revert if slippage array length don't match the token count

✓ should Update Weights and Rebalance (2404ms)

✓ updateTokens should revert if total Weights not equal 10,000 (97ms)

✓ owner should be able to add asset manager

✓ non owner should not be able to add asset manager (49ms)

✓ disable swaphandler in registry should not work if handler array lenght is 0

✓ disable swaphandler in registry should not work if the handler is already disabled

✓ update tokens should not work if the protocol is paused (41ms)

✓ update tokens should not work if swaphandler is not enabled (48ms)

✓ update tokens should not work if non-enabled token is being used (64ms)

✓ update tokens should not work if the function caller is not an asset manager

✓ should get token balance from the rebalance contract

✓ new asset manager should update tokens (2621ms)

✓ withdrawal should revert when contract is paused

✓ should unpause

✓ should pause

✓ should revert unpause

✓ should unpause

✓ when withdraw fund more then balance (43ms)

✓ should fail withdraw when slippage array length is not equal to index length

```
✓ should fail withdraw when balance falls below min investment amount (193ms)
✓ should fail withdraw when balance falls below min investment amount (multi asset) (219ms)
✓ should fail withdraw fund when the output token is not permitted in the asset manager config and is not WETH
✓ should fail withdraw when the protocol is paused (41ms)
✓ should withdraw fund and burn index token successfully (1002ms)
✓ Invest 0.1 WETH into Top10 fund (462ms)
✓ Invest 0.1 WETH into Top10 fund (457ms)
✓ Invest 1 WETH into Top10 fund (433ms)
✓ Invest 1 WETH into Top10 fund (426ms)
✗ should withdraw fund in WETH and burn index token successfully
✓ Invest 1 WETH into Top10 fund (486ms)
✓ should withdraw tokens directly instead of weth (755ms)
✓ should update tokens for more testing (460ms)
✓ should obtain the last rebalance time from the IndexSwap
✓ should introduce a token not having the standard 18 decimals (11700ms)
✓ Invest 1 WETH in the newly rebalanced fund (451ms)
✓ Invest 0.1 WETH should not revert, if investing token is not initialized + tranferFrom should not work (1002ms)
```

→ Tests for MixedIndex - Slippage Control

✓ Should pass the slippage control for small values

✓ Should pass the slippage control for large values

→ Tests for OffChainIndex contract

✓ should add pid (39ms)

✓ Initialize 2nd IndexFund Tokens (102ms)

✓ non-admin should not be able to call the access control setupRole function

✓ admin should be able to call the access control setupRole function

✓ Invest 1 WETH into 1st fund (4704ms)

✓ Invest 2 WETH into Top10 2nd fund (42354ms)

✓ Invest 101.8 USDCe in 1st Index fund (33820ms)

✓ Invest 1 USDCe in 1st Index fund should fail (under min amount) (6147ms)

✓ Invest 50 USDCe should fail, if user input is incorrect in 2nd Index fund (24574ms)

✓ Invest 100 USDCe should fail if user has sent wrong input in 2nd Index fund (20665ms)

✓ Invest 100 USDCe should fail if user tries to manipulate weight in 2nd Index (18989ms)

✓ Invest 1 WETH into 1st Top10 fund (306ms)

✓ Invest 0.1 WETH in 2nd Index fund (31070ms)

✓ Invest 1 WETH into 1st Top10 fund (435ms)

✓ redeem should fail if a non-permitted and non-WETH token is passed as the out asset (77ms)

✓ should withdraw properly with rebalance in between (12256ms)

✓ Invest 1 WETH into 1st Top10 fund (2400ms)

✓ should revert if sellToken address length is manupilated and trigger-multiple withdrawal (11588ms)

✓ Invest 1 WETH into 1st Top10 fund (4704ms)

✓ should Update Weights and Rebalance for 2nd Index (10731ms)

✓ Invest 2 WETH in 2nd Index fund using kyberswap (3704ms)

✓ should swap using handler (700ms)

✓ Invest 2 WETH in 1st Index fund (10876ms)

✓ Investment should fail when the sell token is wrong in the calldata but correct in the contract data (7339ms)

✓ Investment should fail when the sell token is passed as zero address to the invest function (correct calldata, wrong contract data) (5598ms)

✓ Investment should fail when the sell token is manipulated for the invest function (correct calldata, wrong contract data) (6810ms)

✓ Investment should fail when the buy token amount is manipulated in the invest function (correct calldata, wrong contract data) (6429ms)

✓ should fail if offchainHandler is not valid (11045ms)

✓ Invest 1 weth in 1st Index fund should revert if weth value is greater than 0 and investment token is not weth (5571ms)

✓ withdraw should fail if user balance falls below min amount (86ms)

✓ should withdraw fund and burn index token successfully for 1st Index, simultaneously for both user (8588ms)

✓ addr2 should invest using offchain (21534ms)

✓ addr2 should emergency withdraw (8704ms)

✓ owner should invest using offchain (7322ms)

✓ TriggerMultiple TokenWithdrawal withdraw should fail is protocol is paused and work if protocol is unpaused (895ms)

✓ Non owner should not triggerMultiple TokenWithdrawal withdraw

✓ Invest 1 WETH into 1st Top10 fund (889ms)

✓ Withdraw and triggerMultipleWithdrawal should fail if the protocol is paused (6778ms)

→ Tests for priceOracle contract

✓ should revert if aggregator is already added

✓ should revert if base array length does not match the length of other arrays

✓ should revert if quote array length does not match the length of other arrays

✓ should revert if quote array length does not match the length of other arrays

```
✓ Get BTC/USD price (3018ms)
✓ Get BTC/USD price
✓ Get ETH/USD price
✓ Get USDT/USD price
✓ Get DAI/USD price (3361ms)
✓ Get WETH/USD price
✓ Get DOGE/USD price
✓ Get USD/WETH price
✓ Get BTC/WETH price
✓ Get WETH/BTC price
✓ Get ETH/WETH price
✓ Get WETH/ETH price (41ms)
✓ Get DOGE/WETH price (38ms)
✓ Get WETH/DOGE price
✓ Get USD/DOGE price
✓ Get DOGE/weth price
✓ Get weth/DOGE price (41ms)
✓ Get doge/weth price
✓ Get weth/doge price
✓ Get DOGE price in 18 decimals
✓ Get USDT price in 18 decimals
✓ Get ETH price in 18 decimals
✓ Get BTC price in 18 decimals (57ms)
✓ Get WETH_USDT price in 18 decimals (4376ms)
✓ Get WETH_ARB price in 18 decimals (4301ms)
✓ owner updates the oracleTimeout to 35 hours
✓ non owner should not be able to update oracleTimeout
→ ExternalSwapHandler contract
✓ Initialize 1st IndexFund Tokens (61ms)
✓ Initialize 2nd IndexFund Tokens (100ms)
✓ Initialize 3rd IndexFund Tokens (87ms)
✓ Initialize 4th IndexFund Tokens (61ms)
✓ Initialize 5th IndexFund Tokens (59ms)
✓ Initialize 6th IndexFund Tokens (57ms)
✓ Initialize 7th IndexFund Tokens (54ms)
✓ Initialize 8th IndexFund Tokens (53ms)
✓ Invest 0.1ETH into Top10 fund (47676ms)
✓ Invest 0.01ETH into 5th fund (6278ms)
✓ Invest 0.1ETH into 6th fund (4670ms)
✓ Invest 0.2 ETH into index fund (9995ms)
```

✓ Invest 0.2ETH into index fund (2009ms)

✓ Invest 1ETH into Top10 fund (5363ms)

✓ Invest 0.1ETH into Top10 2nd Index fund (1615ms)

✓ Invest 0.1ETH into 7th Index fund (1449ms)

✓ Invest 0.1ETH into 8th index fund (1508ms)

✓ swaps using Kyber Protocol (17387ms)

✓ should revert back if swapHandler is not enabled (613ms)

✓ swaps using 1Inch Protocol (197ms)

✓ revert redeem (113ms)

✓ non assetManager should not revert if 15 minutes is not passed (112ms)

✓ non assetManager should revert if 15 minutes is passed (505ms)

✓ redeems token for 0x (82ms)

✓ swaps reverts if token address is wrong (1687ms)

✓ swaps reverts if sellAmount is wrong (1026ms)

✓ swaps reverts if sellAmount is wrong in calldata (8531ms)

✓ swaps reverts if sellAddress is wrong in calldata (3440ms)

✓ swaps using 0x Protocol (15955ms)

✓ should revert back if the calldata includes fee and the overall slippage is more than 1

✓ update external handler slippage should fail if value is greater than MAX_SLIPPAGE

✓ should update external handler slippage

✓ should set max slippage as 0 and disabling slippage checks (8223ms)

✓ Swaps directly to protocol token DAI and USDCe (6840ms)

✓ Swaps directly to protocol token ERC20 (2994ms)

✓ Invest 0.1ETH into Top10 fund (340ms)

✓ swaps into primary using ZeroEx Protocol from primary (9572ms)

✓ swaps into derivative using ZeroEx Protocol from primary (4352ms)

✓ swaps into lp token using ZeroEx Protocol from primary (8587ms)

✓ Direct Swap reverts if passed underlying token length more than 1 (56ms)

✓ Direct Swap reverts if underlying is not same (58ms)

✓ Direct Swap reverts if length of tokens are not same (49ms)

✓ Direct Swap reverts if length of tokens and sellAmount are not same (45ms)

✓ redeem should revert back if index not paused (548ms)

✓ should pause

✓ redeem should revert back if token getting redeem is not valid

✓ should revert back if the buy token is not registered (79ms)

✓ should revert back if not redeemed (510ms)

✓ should revert back if redeem is called by non asset manager

```
✓ should revert back if metaAggregatorSwap is called by non asset manager (58ms)
✓ Invest 1ETH into Top10 fund (339ms)
→ Tests for Time Dependent contract
✓ Initialize 1st IndexFund Tokens (88ms)
✓ Initialize 2nd IndexFund Tokens (64ms)
✓ Initialize 3rd IndexFund Tokens (60ms)
✓ Initialize 4th IndexFund Tokens (82ms)
✓ Invest 0.11WETH into Top10 fund (12334ms)
✓ Invest 0.1WETH into Top10 2nd index fund (4865ms)
✓ Invest 0.1WETH into Top10 3rd index fund (1145ms)
✓ Invest 0.1WETH into Top10 4th index fund (6840ms)
✓ should revert if the price did not updated for more than 25 hours (111ms)
✓ should revert if the price did not updated for more than 25 hours
✓ should update threshold of the oracle
✓ Asset manager should propose new management fee (44ms)
✓ Asset manager should propose new performance fee (52ms)
✓ Asset manager should propose new entry and exit fee (44ms)
✓ Asset manager should be able to update management fee after 28 days passed (47ms)
```

✓ Asset manager should be able to update performance fee after 28 days passed (39ms)

✓ Asset manager should be able to update entry fee after 28 days passed (42ms)

✓ should claim tokens (3469ms)

✓ should swap reward token using pancakeSwap Handler into derivative token (4895ms)

✓ should claim tokens (844ms)

✓ should swap reward token using pancakeSwap Handler into LP token (779ms)

✓ should claim tokens (848ms)

✓ should claim tokens (921ms)

✓ should swap reward token using pancakeSwap Handler into base token (609ms)

→ Tests for ZeroExSwap contract

✓ Initialize IndexFund Tokens (61ms)

✓ should add pid

✓ should check if off chain handler is enabled or not

✓ Initialize 2nd IndexFund Tokens (73ms)

✓ Invest 1 WETH into Top10 fund (13606ms)

✓ Invest 0.1 WETH into Top10 fund (845ms)

```
✓ Invest 0.1 WETH in first index fund (1425ms)
✓ Should disable external swap handler
✓ update weights should fail if any one weight is zero (86ms)
✓ update weights should fail if sum of weight is not 10000 (63ms)
✓ Update Weights (24186ms)
✓ print values after updating weights to [1000, 2000, 7000] (184ms)
✓ should revert if invalid slippage and _revert after enable Rebalance(1st Transaction) (5591ms)
✓ should _revert after externalSell (2nd Transaction) + revertWithCustomError if InvalidExecution (18409ms)
✓ should update weights (3748ms)
✓ Invest 1 WETH into Top10 fund (352ms)
✓ Invest 1 WETH into Top10 fund (346ms)
✓ Should not update tokens if tokens is not approved
✓ Should not update tokens if tokens is not whitelisted
✓ Should not update if any one weight is zero
✓ Should not update if weight is not equal to 10000
✓ print values before (113ms)
✓ Should Update Tokens (27458ms)
✓ print values after (125ms)
✓ should fail to revert back if all transaction is completed
✓ print values before updating tokens to addresses.WETH (79ms)
✓ should update portfolio to new tokens (2164ms)
✓ print values after updating tokens to addresses.WETH (58ms)
✓ should update tokens (8637ms)
✓ update tokens should fail if update token data is manipulated (3878ms)
✓ update tokens should fail if update token data is manipulated - 02 (4680ms)
✓ Invest 1 WETH into Top10 fund (461ms)
✓ Invest 1 WETH into Top10 fund (463ms)
✓ Invest 1 WETH into Top10 fund (476ms)
✓ Should add one more token (33927ms)
✓ print values after adding one more token ([3000, 1000, 2000, 4000]) (244ms)
✓ Invest 1 WETH into Top10 fund (2148ms)
✓ Should remove one token (6808ms)
✓ Invest 1 WETH into Top10 fund (2344ms)
✓ Should Update Tokens and replace two tokens for vETH and MAIN_LP_BUSD (14175ms)
✓ Invest 1 WETH into Top10 fund (798ms)
✓ should fail if we call wrong revert function (539ms)
✓ print values before reverting (134ms)
```

- ✓ non-assetManager should revert if 15minutes of Pause is passed (4475ms)
- ✓ print values after reverting (129ms)
- ✓ non-assetManager should not be able revert if 15minutes of Pause is not passed (1048ms)
- ✓ it should fail if assetmanager tries to execute 3rd transacton after 1st (4403ms)
- ✓ non assetManager should not be able to update portfolio to new tokens (1796ms)
- ✓ should revert if AlreadyOngoingOperation (709ms)

Coverage:

The code coverage results were obtained by running npx hardhat coverage in the project. We found the following results :

File	% Stmts	% Branch	% Funcs	% Lines
contracts/	79.38	72.73	76	75.7
contracts/access/AccessController.sol	100	100	100	100
contracts/core/	88.32	70.49	91.89	83.64
contracts/fee/	94.05	77.27	95.24	92.05
contracts/handler/	95.35	69.44	83.33	89.25
contracts/handler/Aave/	89.13	45.45	72.73	74.55
contracts/handler/Aave/Interfaces/	100	100	100	100

contracts/handler/ApeSwap/	29.07	16.07	40.91	29.59
contracts/handler/ApeSwap/interfaces/	100	100	100	100
contracts/handler/Beefy/	61.35	28.89	43.33	51.81
contracts/handler/Beefy/interfaces/	100	100	100	100
contracts/handler/BiSwapLP/	0	0	0	0
contracts/handler/BiSwapLP/interfaces/	100	100	100	100
contracts/handler/Compound/	100	55.56	72.73	87.23
contracts/handler/Compound/Interfaces/	100	100	100	100
contracts/handler/ExternalSwapHandler/	34.57	20	60	31.25
contracts/handler/ExternalSwapHandler/Helper/	100	100	100	100
contracts/handler/Hop/	100	57.14	69.23	86.36
contracts/handler/Hop/interfaces/	100	100	100	100
contracts/handler/PancakeSwapLP/	0	0	0	0
contracts/handler/PancakeSwapLP/interfaces/	100	100	100	100
contracts/handler/SushiSwapLP/	86.96	30	58.33	71.43
contracts/handler/SushiSwapLP/interfaces/	100	100	100	100
contracts/handler/Venus/	0	0	0	0
contracts/handler/Venus/interfaces/	100	100	100	100
contracts/handler/Wombat/	87.88	60.71	90.91	81.33
contracts/handler/Wombat/interfaces/	100	100	100	100
contracts/handler/interfaces/	100	100	100	100
contracts/interfaces/	100	100	100	100

contracts/library/	100	50	100	92.31
contracts/oracle/	100	66.67	100	87.5
contracts/oracle/aggregators/	100	50	100	100
contracts/rebalance/	74.3	55.04	82.58	69.59
contracts/registry/	99.53	76.56	98.08	91.04
contracts/vault/	80	50	66.67	80
All files	77.69	55.02	77.15	71.97

6 Conclusion

In this audit, we examined the design and implementation of Velvet Capital V2 - Arbitrum Version contract and discovered several issues of varying severity. Velvet Capital team addressed 2 issues raised in the initial report and implemented the necessary fixes, while classifying the rest as a risk with low-probability of occurrence. Shellboxes' auditors advised Velvet Capital Team to maintain a high level of vigilance and to keep those findings in mind in order to avoid any future complications.

7 Scope Files

7.1 Audit

Files	MD5 Hash
contracts/oracle/PriceOracle.sol	6f25725d355469af4d86e13b8e185c5f
contracts/handler/Hop/HopHandler.sol	9fbf84bdbffb3576ec06c292797df44c
contracts/handler/ExternalSwapHandler/BebopHandler.sol	e0157c223120d032a39c3a67ddaa7d68
contracts/handler/ExternalSwapHandler/KyberSwapHandler.sol	a957efe6d2c4e85ba12b1d2446f5c62c
contracts/handler/Compound/CompoundV3Handler.sol	0bff92e30a678fa209fb69eb0fab6de2
contracts/handler/Beefy/BeefyBridgeHandler.sol	bc2b710f570e920841a9be1a6e84f7e6
contracts/handler/Aave/AaveV3Handler.sol	e80ab617b4d317d1f77d572aa3039236

7.2 Re-Audit

Files	MD5 Hash
contracts/oracle/PriceOracleL2.sol	40003af800e16ea680ca8ded1e6b181b
contracts/handler/Hop/HopHandler.sol	9fbf84bdbffb3576ec06c292797df44c
contracts/handler/ExternalSwapHandler/BebopHandler.sol	1064c3ca1ce3632db013a2f75859da70
contracts/handler/ExternalSwapHandler/KyberSwapHandler.sol	65b1193d7e28181e87444292f68d241c

contracts/handler/Compound/CompoundV3Handler.sol	26877a97d77a1fadf4ebdca598f4e428
contracts/handler/Beefy/BeefyBridgeHandler.sol	fb911902f0e9f90f3088f5afcf4ca55b
contracts/handler/Aave/AaveV3Handler.sol	2beacfb3f1209faff42056a9c70ae516

8 Disclaimer

Shellboxes reports should not be construed as ”endorsements” or ”disapprovals” of particular teams or projects. These reports do not reflect the economics or value of any ”product” or ”asset” produced by any team or project that engages Shellboxes to do a security evaluation, nor should they be regarded as such. Shellboxes Reports do not provide any warranty or guarantee regarding the absolute bug-free nature of the examined technology, nor do they provide any indication of the technology’s proprietors, business model, business or legal compliance. Shellboxes Reports should not be used in any way to decide whether to invest in or take part in a certain project. These reports don’t offer any kind of investing advice and shouldn’t be used that way. Shellboxes Reports are the result of a thorough auditing process designed to assist our clients in improving the quality of their code while lowering the significant risk posed by blockchain technology. According to Shellboxes, each business and person is in charge of their own due diligence and ongoing security. Shellboxes does not guarantee the security or functionality of the technology we agree to research; instead, our purpose is to assist in limiting the attack vectors and the high degree of variation associated with using new and evolving technologies.

SHELLBOXES

For a Contract Audit, contact us at contact@shellboxes.com