Gearbox Finance V2

Date	August 2022
Auditors	David Oz Kashi, Tejaswa Rastogi, David Braun

1 Executive Summary

This report presents the results of our engagement with **Gearbox** to review **Gearbox v2 smart contracts**.

The review was conducted over three weeks, from **July 25, 2022** to **August 12, 2022** by **David Oz Kashi**, **Tejaswa Rastogi** and **David Braun**. A total of 45 person-days were spent.

2 Scope

Our review focused on the commit hash `8ea754fafa5ada5b099b1905dca8d05d5c22078a`. The list of files in scope can be found in the Appendix.

Note: The original `contracts-v2` repository has been split into multiple repositories.
Core contracts and supporting infrastructure are moved to `core-v2: c6ca919d46dcd82fa69c89316d9ff969e89bd3f6` and

Contracts related to integration with third-party protocols are moved to `integrations-v2: 76d3733b7183da3deca51b212a959dfdc622e96d`

The repositories contains no Major difference with the `contracts-v2` repository. However, we recommend updating the test suite to thoroughly test and verify the minor adjustments done.

2.1 Objectives

Together with the **Gearbox** team, we identified the following priorities for our review:

1. Ensure that the system is implemented consistently with the intended functionality, and without unintended edge cases.
2. Identify known vulnerabilities particular to smart contract systems, as outlined in our Smart Contract Best Practices, and the Smart Contract Weakness Classification Registry.

3 Recommendations

3.1 Remove `hardhat/console.sol` imports ✓ Fixed

Resolution

The instances of `hardhat/console.sol` have been removed in a pull request with final commit hash `4e5e37d2309a823b87b0d4e7de9b9b1142f0e15c`

Description

The `hardhat/console.sol` module is used for debugging purposes and should not exist in production code.

Examples

- `CurveV1_Base.sol`
- `CreditConfigurator.sol`
- `CreditFacade.sol`
- `CreditManager.sol`
- `CreditManagerFactory.sol`
- `PriceOracle.sol`
- `UniswapV2PathFinder.sol`

3.2 `CreditFacade._wrapETH` should revert for deposits of both ETH and tokens Acknowledged

Resolution

Comment from Gearbox:
The purpose of `CreditFacade._wrapETH` is to convert any ETH sent to Credit Facade into WETH, in preparation for further actions (such as adding WETH as collateral). Gearbox V2 does not support operations with native ETH and only allows WETH as collateral.

As such, whether any other tokens participate in a transaction is not particularly relevant - the function is there simply to allow users that only have ETH to interact with contracts, without explicitly calling WETH.deposit() first. Under no circumstances does Gearbox V2 accept native ETH.

Description

`CreditFacade._wrapETH` should receive either ETH (exclusive) or tokens.

Recommendation

Consider using the `amount` argument to determine whether it's a tokens deposit, and if so, revert for `msg.value > 0`.

3.3 Consider using `SafeCast` ✓ Fixed

Resolution

Instances of unsafe downcasting in `CurveV1_Base` and `CreditManager` have been replaced with suggested SafeCast's safe downcasting functions in a pull request with final commit hash `4e5e37d2309a823b87b0d4e7de9b9b1142f0e15c`.

Comment from Gearbox:
One notable exception is `CreditManager.collateralTokensByMask()`. Values in `collateralTokensCompressed` encode the token address as the first 160 bits and the token liquidation threshold to the left of that. The standard behavior when downcasting a large number (any extra bits to the left of the new width are discarded) is used to extract the encoded address from the first 160 bits of the unit256 number.

```
token = address(uint160(collateralTokenCompressed));
liquidationThreshold = uint16(
    collateralTokenCompressed >> ADDR_BIT_SIZE
);
```

Description

Downcasting operations were found in various places in the codebase. We were not able to find any concrete exploitable scenarios, however, it is recommended to use the `SafeCast` library, or alternatively, to add inline comments which prove that unintended scenarios are impossible.

3.4 Refactor `CreditFacade._multicall` ✓ Fixed

Resolution

The function `_multicall` has now been refactored into `_multicall` and `_processCreditFacadeMulticall`. The latter now handles calls for `CreditFacade` itself. The changes were implemented in a pull request with final commit `4e5e37d2309a823b87b0d4e7de9b9b1142f0e15c`

Description

The `_multicall` method of `CreditFacade` is security sensitive and 125 lines long. Small functions composed together are easier to read and to reason about which is especially important in security-critical software.

Recommendation

Refactor the method in accordance with the best practice of sizing a function to fit within a typical computer screen.

3.5 Gas optimizations Partially Addressed

Resolution

Comment from Gearbox (for Optimization#1):
This optimization was not implemented, since this would require either additional parameters introduced to relevant functions, or changing the control flow.

Though the post-increment operators have now been replaced with pre-increment operators.

Description

1. The main flow of `CreditFacade.openCreditAccount` is calling `_revertIfActionOnAccountNotAllowed` twice instead of once.
2. Number increments should use the pre-increment operator consistently throughout the codebase.

3.6 Several contracts allow ownership transfer in single step Partially Addressed

Resolution

`ACL` and `AddressProvider` are now following the suggested two-step approach to transfer Ownership via `Claimable` contract. However, `GearboxToken` still follows the same traditional approach, which we recommend modifying as well.

Also, we recommend overriding the `renounceOwnership` to renounce Pending Owner as well. As it may happen that even though the ownership had been renounced, there may still exist a pending owner which can claim back the ownership again.

The changes are implemented in a pull request with final commit as `4e5e37d2309a823b87b0d4e7de9b9b1142f0e15c`

Description

Several contracts in the codebase provide a `transferOwnership` method that allows changing the manager of the contract in a single step. This high-stakes operation offers no recourse if there is a mistake in the new manager address, leading to a situation in which the contract may have no functional manager or a malicious one. The risk of the single-step process in OpenZeppelin's Ownable library, used by many of the contracts, has been publicly discussed.

Examples

code/contracts/core/ACL.sol:L12

```
contract ACL is Ownable, IACL {
```

code/contracts/core/AddressProvider.sol:L23

```
contract AddressProvider is Ownable, IAddressProvider {
```

code/contracts/tokens/DieselToken.sol:L10

```
contract DieselToken is ERC20, Ownable {
```

code/contracts/tokens/GearToken.sol:L116-L123

```
function transferOwnership(address newManager)
    external
    managerOnly // T:[GT-3]
{
    require(newManager != address(0), "Zero address is not allowed"); // T:[GT-5]
    emit OwnershipTransferred(manager, newManager); // T:[GT-6]
    manager = newManager; // T:[GT-6]
}
```

Recommendation

We recommend using a two-step pattern in which a new manager is proposed by the current one and the pending manager is required to claim the new ownership.

3.7 Consider informing users about the historic allowances of their credit account Acknowledged

Resolution

Comment from Gearbox:

"Safe account opening" will be an option when opening an account through the Gearbox V2 frontend. When the user opens an account with the option enabled, allowances of the current head in the Credit Account factory are retrieved, and the necessary revocation calls are added to the account opening multicall.

Description

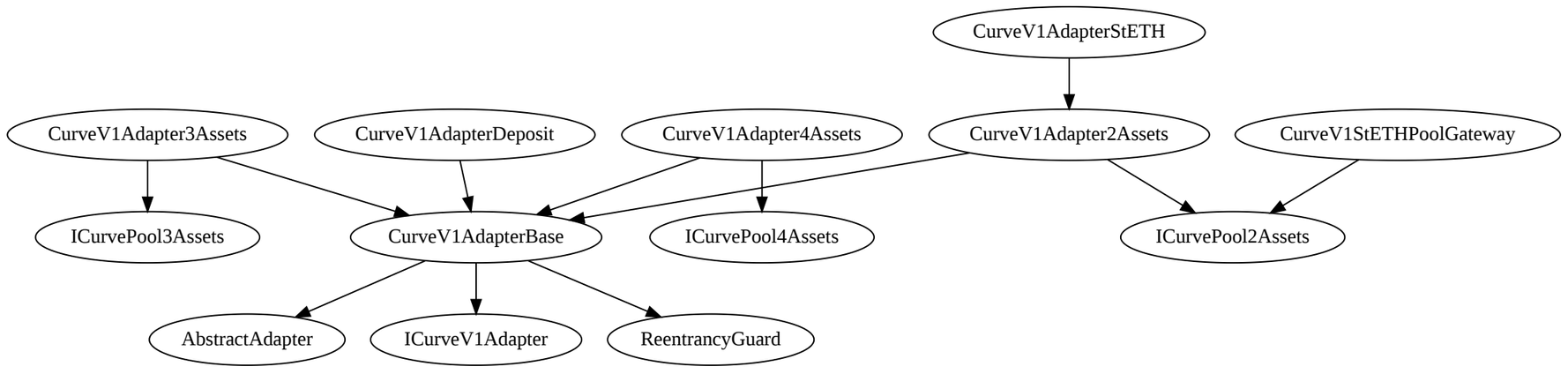
Credit accounts are reused between different users in the system. When a user opens a new credit account he might get an already existing one that was previously used by someone else. `UniversalAdapter` allows a credit account owner to revoke any allowance that was given in the past, but credit account owners might not be aware of that. In general, having unwanted allowances is not necessarily considered a security issue, but it increases the potential attack surface and might amplify the damage of a security issue that was found in external contracts.

Recommendation

Consider adding a page in the web application graphical interface that allows the credit account owner to manage the allowances given to the credit account he currently owns.

4 System References

4.1 Curve Adapter Inheritance



4.2 Adapter Integrations

The following is a list of state-changing integration functions called by adapter code (with links to documentation) that were checked during the audit.

- Convex
  - BaseRewardPool
    - getReward
    - stake
    - withdraw
    - withdrawAll
    - withdrawAllAndUnwrap
    - withdrawAndUnwrap
  - Booster
    - deposit
    - depositAll
    - withdraw
    - withdrawAll
  - ClaimZap
    - claimRewards
- Curve
  - StableSwap
    - add_liquidity
    - exchange_underlying
    - exchange
    - remove_liquidity
    - remove_liquidity_imbalance
    - remove_liquidity_one_coin
- Lido
  - submit
- Uniswap
  - V2
    - swapExactTokensForTokens
    - swapTokensForExactTokens
  - V3
    - exactInput
    - exactInputSingle
    - exactOutput
    - exactOutputSingle
- Yearn
  - deposit
  - withdraw
  - withdraw(uint256,address,uint256)

5 Findings

Each issue has an assigned severity:

- Minor issues are subjective in nature. They are typically suggestions around best practices or readability. Code maintainers should use their own judgment as to whether to address such issues.
- Medium issues are objective in nature but are not security vulnerabilities. These should be addressed unless there is a clear reason not to.
- Major 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.
- Critical issues are directly exploitable security vulnerabilities that need to be fixed.

5.1 Attackers can open credit accounts that will never be liquidated Major ✓ Fixed

Resolution

`CreditManager._checkAndOptimizeEnabledTokens()` has been implemented to limit enabled tokens per Credit Account to 12, which can be changed whenever needed. As per Gearbox, the number is based on internal gas profiling and estimated usage scenarios. However, the number is not verified by the audit team, and we recommend handling it with care as per the requirements.

This function is called in cases where new tokens can be enabled, for instance:

1. In `fastCollateralCheck`, as it is invoked at the end of an adapter call.
2. In `fullCollateralCheck`, as it is invoked at the end of an adapter call or a Credit Facade multicall.
3. In `CreditFacade.addCollateral()` and `CreditFacade.enableToken()`.

The changes are implemented in a pull request with final commit as `4e5e37d2309a823b87b0d4e7de9b9b1142f0e15c`

Description

In theory, a credit account can be liquidated by anyone once the health factor drops below 1. In practice, however, liquidations are transactions executed by the EVM, which means liquidation transactions that consume more gas than the block gas limit will fail, thus the liquidation will never occur. Currently, the gas amount for a liquidation transaction increases in direct proportion to the number of different tokens used as collateral in the credit account. Attackers can open credit accounts with multiple tokens as collateral, which will cause any attempt of future liquidation to fail due to out of gas exception.

Recommendation

Consider running gas profiling tests to measure the maximum number of different tokens possible. Use this number to limit the allowed number of different tokens that can be used as collateral for a credit account.

5.2 Over-Reliance on Single Oracle Medium Acknowledged

Resolution

Comment from Gearbox:
The concern is valid, however, this change would require a major rewrite of PriceOracle and adjacent contracts, as well as deployment scripts and other middleware. Additional research would also need to be performed, in order to determine reliable price feed providers outside of Chainlink.

Gearbox team acknowledges the issue and will be revisiting it at a later date

Description

Gearbox uses Chainlink to determine asset prices when computing the health factor of a **Credit Account**. Relying on a single oracle for price data is risky because the oracle could be compromised or faulty (e.g., the Synthetix incident in which a bot was able to exploit bad price data to generate USD 1 billion in profit).

Recommendation

We recommend sampling price data from multiple oracles and discarding any outlier values to protect against bad oracle data.

5.3 DegenNFT: token ID collision may lead to DoS Minor ✓ Fixed

Resolution

The logic has been modified as:

```
    uint256 tokenId = (uint256(uint160(to)) << 40) + balanceBefore + i;
    _mint(to, tokenId);
```

which adds a `2^40` uint buffer space for every address salt to mint token IDs, which would be practically sufficient to avoid token ID collisions. However, theoretically, the vector still exists.

Description

The contract uses the destination address as salt, to mint the desired number of token IDs. However, there is a possibility/risk of collision. A token ID, which is subject to be minted for an address, may collide with an existing token ID.

For instance, let's assume: `address X = 0x5b38da6A701C568545DcFcB03FcB875f56BEddc9` holds 1 DegenNFT **token ID** will be the address itself, i.e, `520786028573371803640530888255888666801131675081` in decimals

Now, the configurator has to mint 6 DegenNFT to

`address Y = 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4` adding 6 starting from the address itself will generate the same token ID that has already been minted to address X, and hence will be reverted. It means it is not possible to mint more DegenNFTs to address Y, until and unless the token ID allocated to address X is burned.

It can be even used to deliberately disallow minting for a particular account.

Examples

code/contracts/tokens/DegenNFT.sol:L126-L143

```
function mint(address to, uint256 amount)
    external
    override
    onlyMinter // F:[DNFT-3]
{
    uint256 balanceBefore = balanceOf(to); // F:[DNFT-7]

    for (uint256 i; i < amount; ) {
        uint256 tokenId = uint256(uint160(to)) + balanceBefore + i; // F:[DNFT-7]
        _mint(to, tokenId); // F:[DNFT-7]

        unchecked {
            ++i; // F:[DNFT-7]
        }
    }

    totalSupply += amount; // F:[DNFT-7]
}
```

Recommendation

A better approach could be to use an increasing counter to avoid collisions. An example approach could be to use `Counters` library from Openzeppelin

```
...
import "@openzeppelin/.../Counters.sol";
contract DegenNFT is ERC721, ACLTrait, IDegenNFT {
...
using Counters for Counters.Counter;
Counters.Counter private _tokenIdCounter;

function mint(address to, uint256 amount) external override onlyMinter {
...
uint256 tokenId = _tokenIdCounter.current();
_tokenIdCounter.increment();
_mint(to, tokenId);

      }

}
```

5.4 ERC-20 `approve` method called with maximum amount value Minor Acknowledged

Resolution

Comment from Gearbox:
The Gearbox team ultimately made a choice to leave the feature as is. Max allowances are required to provide adequate user experience when interacting with Uniswap or Curve, since, without max allowances, approving tokens would be required for each trade. Max allowances are only permitted for highly-trusted and immutable protocols, such as Uniswap and Curve. For other protocols, especially upgradeable ones, allowances are never more than 1 between transactions.

Though we still recommend following the best practice, i.e, not to provide max allowance for tokens, so as to avoid any edge case scenarios.

Description

The contract adapters allow users to place trades, stake tokens, and to provide liquidity using ERC-20 tokens in credit accounts. Before each transaction, the ERC-20 protocol requires a credit account to approve the amount to be transferred by the third-party contract. In multiple instances of the code the approval is made with the maximum possible value. This shortcut leaves open the possibility that malicious code could transfer more than the intended number of tokens. While the probability of this is low with well-vetted, high-profile third party contracts, it's an opening that potentially could be combined with a possible flaw elsewhere in the code to steal funds.

Examples

code/contracts/adapters/AbstractAdapter.sol:L56-L61

```
creditManager.approveCreditAccount(
    msg.sender,
    targetContract,
    tokenIn,
    type(uint256).max
);
```

code/contracts/adapters/AbstractAdapter.sol:L71-L76

```
creditManager.approveCreditAccount(
    msg.sender,
    targetContract,
    tokenIn,
    type(uint256).max
);
```

code/contracts/adapters/AbstractAdapter.sol:L129-L134

```
creditManager.approveCreditAccount(
    msg.sender,
    targetContract,
    tokenIn,
    type(uint256).max
);
```

code/contracts/adapters/curve/CurveV1_Base.sol:L806-L811

```
creditManager.approveCreditAccount(
    msg.sender,
    targetContract,
    token0,
    type(uint256).max
); // F:[ACV1_2-4, ACV1_3-4, ACV1_4-4]
```

code/contracts/adapters/curve/CurveV1_Base.sol:L814-L819

```
creditManager.approveCreditAccount(
    msg.sender,
    targetContract,
    token1,
    type(uint256).max
); // F:[ACV1_2-4, ACV1_3-4, ACV1_4-4]
```

code/contracts/adapters/curve/CurveV1_Base.sol:L822-L827

```
creditManager.approveCreditAccount(
    msg.sender,
    targetContract,
    token2,
    type(uint256).max
); // F:[ACV1_3-4, ACV1_4-4]
```

code/contracts/adapters/curve/CurveV1_Base.sol:L830-L835

```
creditManager.approveCreditAccount(
    msg.sender,
    targetContract,
    token3,
    type(uint256).max
); // F:[ACV1_4-4]
```

code/contracts/adapters/curve/CurveV1_DepositZap.sol:L31-L36

```
creditManager.approveCreditAccount(
    msg.sender,
    targetContract,
    lp_token,
    type(uint256).max
);
```

code/contracts/adapters/curve/CurveV1_DepositZap.sol:L38-L43

```
creditManager.approveCreditAccount(
    msg.sender,
    targetContract,
    lp_token,
    type(uint256).max
);
```

code/contracts/adapters/curve/CurveV1_stETH.sol:L55-L60

```
creditManager.approveCreditAccount(
    msg.sender,
    targetContract,
    lp_token,
    type(uint256).max
);
```

code/contracts/adapters/curve/CurveV1_stETH.sol:L62-L67

```
creditManager.approveCreditAccount(
    msg.sender,
    targetContract,
    lp_token,
    type(uint256).max
);
```

code/contracts/adapters/curve/CurveV1_stETHGateway.sol:L61

```
IERC20(token1).approve(pool, type(uint256).max);
```

Recommendation

In virtually every case the amount of tokens to be transferred is known prior to the transaction. We recommend that the precise amount to be transferred is used in the ERC-20 `approve` call.

5.5 ACL - Lack of Existing Address Checks may make off-chain monitoring difficult or trigger False Alarms ✓ Fixed

Resolution

The issue has been fixed in a pull request with a final commit hash `4e5e37d2309a823b87b0d4e7de9b9b1142f0e15c`. The concerned functions `removePausableAdmin` and `removeUnpausableAdmin` now revert when the passed address is not in the respective set.

Description

ACL defines two privileged roles set `pausableAdminSet` and `unpausableAdminSet`. The privileged roles as the name suggests can pause/unpause contracts. The contract defines `onlyOwner` functions to add/remove accounts from these sets. They also emit events to log the addresses added/removed from these sets. However, they don't validate the address passed as a parameter, which may lead to false alarms or make off-chain monitoring difficult.

For Instance: Address X, never belonged to `pausableAdminSet` or `unpausableAdminSet`. Still, there is a possibility, that `removePausableAdmin` / `removeUnpausableAdmin` gets called for the address. It will emit the event thereby logging the address, and making a notification for the off-chain monitoring tools. It may create a panic, forcing the team to reevaluate the past transactions, questioning, "Why/When this address had been made a pausable/unpausable admin?"

Examples

code/contracts/core/ACL.sol:L31-L37

```
function removePausableAdmin(address admin)
    external
    onlyOwner // T:[ACL-1]
{
    pausableAdminSet[admin] = false; // T:[ACL-3]
    emit PausableAdminRemoved(admin); // T:[ACL-3]
}
```

Recommendation

Address checks may be added to ensure that only existing admins are removed.

```
    function removePausableAdmin(address admin)
        external
        onlyOwner // T:[ACL-1]
    {   require(pausableAdminSet[admin], "Not an Admin");
        pausableAdminSet[admin] = false; // T:[ACL-3]
        emit PausableAdminRemoved(admin); // T:[ACL-3]
    }
```

6 Security Specification

This section describes some risks, and trust assumptions that are consequences of its design.

6.1 Privileged users and trust assumptions

Privileged users are trusted by the users and has unilateral control over the system. Among other actions, this role may without a warning, change important parameters that govern essential parts of the contract's behavior, and these changes take effect immediately.

The system's security heavily relies on the security of the integrated third parties. It is assumed that defi protocols and tokens exhibit correct and consistent behaviour, e.g. in terms of token balances and transfers. Therefore, we strongly recommend the implementation of a thorough vetting process before a new token or an adapter is integrated into the Gearbox system.

A checklist for an integration candidate could include points such as:

- Reviewing audit report(s) performed by well-known, independent professionals
- Reviewing the adherence to standard interfaces
- Ensuring that security-critical components are not upgradable
- Ensuring that the candidate's development team has a process in place to handle security incidents
- Ensuring that a point of contact is known to handle potential legal issues
- In the case of new adapters, ensuring that fast checks are sufficient to avoid any sort of attack that will lead to loss of funds.

6.2 Additional observations

Update: The observations have been addressed in a pull request with a final commit hash `4e5e37d2309a823b87b0d4e7de9b9b1142f0e15c`

1. Variables in `CreditManager` has now been rearranged, and the concerned variable has now been renamed and accurately lies in slot#1.
2. `PoolServicenow` sets `_timestampLU` to block.timestamp in the constructor.
3. `WadRayMath` now has been removed.

- `CreditManager` has a variable named `slot0` which in practice is stored in slot#2.

- The current smart contract architecture results in many duplicated contracts. New adapters will have to be deployed for any `CreditManager` that was previously deployed, since `CreditManager` only supports a single underlying. It will make the process of managing and monitoring the Gearbox deployment more complex and inefficient in terms of gas.
- `PoolService` at the time of initalization, updates borrow rate with function `_updateBorrowRate`. Expected liquidity and cumulative index calculation depends upon the timedifference as `block.timestamp - _timestampLU`, where `_timestampLU` represents the last updated time. However, `_timestampLU` is not initalized yet and points to 0, which means it calculates a huge time difference. As there is no borrowed amount and borrow rate at the time of initialization, it's not affecting the expected liquidity and cumulative index calculation(the resultant multiplication will be 0). But if the team plans to deploy the contract with some predefined borrowed amount and rate, the contract will generate accrued interest from ghost blocks.
- `WadRayMath` performs overflow/underflow checks, for instance, `require(a <= (type(uint256).max - halfWAD) / b,Errors.MATH_MULTIPLICATION_OVERFLOW);` which is costing unnecessary gas, as the compiler does that on its own.

Appendix 1 - Files in Scope

This audit covered the following files:

File Name	SHA-1 Hash
/contracts/adapters/AbstractAdapter.sol	`76d94692da02d39818533ad0469b2aa764b67556`
/contracts/adapters/UniversalAdapter.sol	`bd92df24a19d71f7e401c01423a248a905a3c4a1`
/contracts/adapters/convex/ConvexV1_BaseRewardPool.sol	`cf9a7c33fb920291988643a0bb862b64345ea232`
/contracts/adapters/convex/ConvexV1_Booster.sol	`5a538648fd8244a3c056a5195cfc6464b182ff70`
/contracts/adapters/convex/ConvexV1_ClaimZap.sol	`a8ce6a62b6c824adfd8a4655aa7792190aa1266f`
/contracts/adapters/convex/ConvexV1_StakedPositionToken.sol	`acc72eb26b48114db83aba7e69fb166c5cad7432`
/contracts/adapters/curve/CurveV1_2.sol	`d36a0436ffc1e5bceeb4f87180de9f7fae867150`
/contracts/adapters/curve/CurveV1_3.sol	`7259158045519c3c8e932bac57db3fefc48ab0dd`
/contracts/adapters/curve/CurveV1_4.sol	`06927a04de72fa4f5b63175409ac90f4eee7f8cf`
/contracts/adapters/curve/CurveV1_Base.sol	`7dd08fd4899ccd5fc3d6260dd331f6622750fb5e`
/contracts/adapters/curve/CurveV1_DepositZap.sol	`e19e5cc16b681c38cfef087c17377f34d31f5c5a`
/contracts/adapters/curve/CurveV1_stETH.sol	`595aba5c5a8e5e92ade420d285316e17a46e7efc`
/contracts/adapters/curve/CurveV1_stETHGateway.sol	`03b9ed5a11ded74041c15f8b5398f0dd6623723d`
/contracts/adapters/lido/LidoV1.sol	`82c7949a6dad224353a8deea67db8af278b4ac24`
/contracts/adapters/lido/LidoV1_WETHGateway.sol	`87b5350c9d40d2aaf29c934826c249061f419dbe`
/contracts/adapters/uniswap/UniswapV2.sol	`f948f1e2f443b847c5b2d3d76a8cea58d648f6ad`
/contracts/adapters/uniswap/UniswapV3.sol	`41248d5a1313be6a27b778ec6a2820e621f2ee60`
/contracts/adapters/yearn/YearnV2.sol	`8678ce990f9ee6e5bd63d82365f673f52c70466c`
/contracts/core/ACL.sol	`b9366e6c05f532f91ac68534a90efdfe6b6688a3`
/contracts/core/ACLTrait.sol	`ff9e7945c7cef849fb64859f38e5de453f16fd48`
/contracts/core/AccountFactory.sol	`9c786c096088837b5a83ccaaa32874f3dd0abcb0`
/contracts/core/AddressProvider.sol	`1e90baff6fc50465ca7163ccc9db2a7c6b93533c`
/contracts/core/WETHGateway.sol	`7a7a5acff6b0c08e701c49ab4d3f744ae1840c33`
/contracts/credit/CreditAccount.sol	`4ca7f57f98b3a3f4ea4b859efe02d5745b696e94`
/contracts/credit/CreditConfigurator.sol	`3d5fce142f76c470faa03e488c9f70294bdadc22`
/contracts/credit/CreditFacade.sol	`fc15e60d625ec1344c711d47c58f6abe25289975`
/contracts/credit/CreditManager.sol	`8157b189bfac0cce0e8ce539678433bd85503ba5`
/contracts/interfaces/IACL.sol	`e1466c08f205deef3eaa6eeeba17725970832fc5`
/contracts/interfaces/IAccountFactory.sol	`2890fd84155f60874ce804715cb0b108b03ecaed`
/contracts/interfaces/IAddressProvider.sol	`7513fc16a5ce1752cbb33436ad233e00a04f35de`
/contracts/interfaces/IContractsRegister.sol	`e1941fa073c3ce6b244b124f3dbb3e1efbe0785c`
/contracts/interfaces/ICreditAccount.sol	`5dff713d26736cfc4224a22a5b08ed148c5b1d7a`
/contracts/interfaces/ICreditConfigurator.sol	`43a54a234aa06e6870272371a33694a2d158b4ce`
/contracts/interfaces/ICreditFacade.sol	`3ed45c230f3eecefcf69b793fdc73788ea99e07c`
/contracts/interfaces/ICreditManagerV2.sol	`bcaff57e21aed685109d86eb029323d0153f2bd7`
/contracts/interfaces/IDataCompressor.sol	`f1402e261a7a6723cc84b69e3f4b12db3b166a47`
/contracts/interfaces/IDegenNFT.sol	`2c692a1dcea2fa9698ca900a3ac1dbdb0423247c`
/contracts/interfaces/IErrors.sol	`fbcc0dc86d4fb47566d420bb869660146aef8c91`
/contracts/interfaces/IGearToken.sol	`8c5e3617851810b78017208351b4938831c73572`
/contracts/interfaces/IInterestRateModel.sol	`6530d7d7a847826b84040d2a4320991cdd4426a7`
/contracts/interfaces/ILPPriceFeed.sol	`9bd2e944848184f9cf399afb08c4447fceaf6bbd`

File Name	SHA-1 Hash
/contracts/interfaces/IMerkleDistributor.sol	`cee03b111016d355e6f6dcaf225ae8a2b055835b`
/contracts/interfaces/IPhantomERC20.sol	`df5021f0b3005234ee779363146bd8c4dffc5519`
/contracts/interfaces/IPoolService.sol	`10f0bb4d36f2d88b8f02e1577358edc7f372eca3`
/contracts/interfaces/IPriceFeedAddress.sol	`afdd9c80c8536573553ef2311de54d1bdf6ad46b`
/contracts/interfaces/IPriceFeedType.sol	`f9266cf11df874cebc8ebb987336048ea599fac7`
/contracts/interfaces/IPriceOracle.sol	`a713ddf0836d0a881f097977ee67becd0976d5a6`
/contracts/interfaces/IVersion.sol	`4b95719fed8311e1bb18b442d81d17ccce36e863`
/contracts/interfaces/IWETHGateway.sol	`ad46c4e16f9fc1b04b95233e85b104513e391dba`
/contracts/interfaces/V1/ICreditFilter.sol	`49434901b9aee9328582d3297ae3bd302a8560a0`
/contracts/interfaces/V1/ICreditManager.sol	`d5005761e6461dc72ab5d6db9027bd433f4b9114`
/contracts/interfaces/V1/IPriceOracle.sol	`54ae7f871a63cd631519e5a3a0a624c05099780e`
/contracts/interfaces/adapters/IAdapter.sol	`a9249ecd470b326bcd15f63a0191454954919957`
/contracts/interfaces/adapters/IUniversalAdapter.sol	`9cfed1304a0ae040c59cf667ed2e92920195d3a7`
/contracts/interfaces/adapters/convex/IConvexV1BaseRewardPoolAdapter.sol	`22646add0729aa9be7ab0252f6f17a2942622125`
/contracts/interfaces/adapters/convex/IConvexV1BoosterAdapter.sol	`9615978d93ffe58d4ee8bafc9b739e0796074cdb`
/contracts/interfaces/adapters/curve/ICurveV1Adapter.sol	`1b8a191a9f86a9f81a702950979a915efd3eb470`
/contracts/interfaces/adapters/lido/ILidoV1Adapter.sol	`3766fde961f875c6a283c880141eba0469e78eaa`
/contracts/interfaces/adapters/uniswap/IUniswapV2Adapter.sol	`733523892d87bd9e19a4127d87b75262239e2023`
/contracts/interfaces/adapters/uniswap/IUniswapV3Adapter.sol	`a08db347d1db073fa751de204307e0d5a79e78cc`
/contracts/interfaces/external/IWETH.sol	`b6993ada5f2e2b87d70fee0cbb1ad9355724879b`
/contracts/libraries/Errors.sol	`dd82d6912864968c2877596d1746c8449f305b08`
/contracts/oracles/LPPriceFeed.sol	`3c5b7ecf67b5292fba4eaf041eea359b1c23e2cd`
/contracts/oracles/PriceFeedChecker.sol	`d86854eb731ffcaee4bc6489ea5ec5c2f7f0b803`
/contracts/oracles/PriceOracle.sol	`fb711e3307065cdce4504d667787a69a6b5d6f28`
/contracts/oracles/ZeroPriceFeed.sol	`571b97cb64ba6923962c1e95f36ffc8f83dddfac`
/contracts/oracles/curve/AbstractCurveLPPriceFeed.sol	`aafdaba35c4da30154b88cef23f2679147009e47`
/contracts/oracles/curve/CurveLP2PriceFeed.sol	`0c77748006bdcc2f2f4eb388c5f8e31a2d5b5022`
/contracts/oracles/curve/CurveLP3PriceFeed.sol	`eab14dc21c8d8c1a7e7f8a85eb5534c92b0021b6`
/contracts/oracles/curve/CurveLP4PriceFeed.sol	`19e3cd84cf6671360ff0dda45ba749a852a96375`
/contracts/oracles/yearn/YearnPriceFeed.sol	`67ad71fe65dcce359d9fa0fc0036e45cf0cff963`

Appendix 2 - Disclosure

ConsenSys Diligence (“CD”) typically receives compensation from one or more clients (the “Clients”) for performing the analysis contained in these reports (the “Reports”). The Reports may be distributed through other means, including via ConsenSys publications and other distributions.

The Reports are not an endorsement or indictment of any particular project or team, and the Reports do not guarantee the security of any particular project. This Report does not consider, and should not be interpreted as considering or having any bearing on, the potential economics of a token, token sale or any other product, service or other asset. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. No Report provides any warranty or representation to any Third-Party in any respect, including regarding the bugfree nature of code, the business model or proprietors of any such business model, and the legal compliance of any such business. No third party should rely on the Reports in any way, including for the purpose of making any decisions to buy or sell any token, product, service or other asset. Specifically, for the avoidance of doubt, this Report does not constitute investment advice, is not an endorsement of this project or team, and it is not a guarantee as to the absolute security of the project. CD owes no duty to any Third-Party by virtue of publishing these Reports.

PURPOSE OF REPORTS The Reports and the analysis described therein are created solely for Clients and published with their consent. The scope of our review is limited to a review of code and only the code we note as being within the scope of our review within this report. Any Solidity code itself presents unique and unquantifiable risks as the Solidity language itself remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond specified code that could present security risks. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. In some instances, we may perform penetration testing or infrastructure assessments depending on the scope of the particular engagement.

CD makes the Reports available to parties other than the Clients (i.e., “third parties”) – on its website. CD hopes that by making these analyses publicly available, it can help the blockchain ecosystem develop technical best practices in this rapidly evolving area of innovation.

LINKS TO OTHER WEB SITES FROM THIS WEB SITE You may, through hypertext or other computer links, gain access to web sites operated by persons other than ConsenSys and CD. Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites’ owners. You agree that ConsenSys and CD are not responsible for the content or operation of such Web sites, and that ConsenSys and CD shall have no liability to you or any other person or entity for the use of third party Web sites. Except as described below, a hyperlink from this web Site to another web site does not imply or

mean that ConsenSys and CD endorses the content on that Web site or the operator or operations of that site. You are solely responsible for determining the extent to which you may use any content at any other web sites to which you link from the Reports. ConsenSys and CD assumes no responsibility for the use of third party software on the Web Site and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any outcome generated by such software.

TIMELINESS OF CONTENT The content contained in the Reports is current as of the date appearing on the Report and is subject to change without notice. Unless indicated otherwise, by ConsenSys and CD.