feeBeraborrow Core Audit Documentation

Overview

What is Beraborrow?

Beraborrow is a CDP featuring stablecoin loans backed by native Berachain assets, ETH LSTs, BTC derivatives, yield-bearing stablecoins and Proof of Liquidity assets (iBGT & iBERA).

Based on Prisma Finance architecture (initial fork was done against this commit), which is based on Liquity's, but enabling multi-collateral support and more control over key collateral parameters.

Introduce a novel tokenized Stability Pool and collaterals are wrapped on CollateralVaults, which makes your collaterals further yield bearing by autocompounding PoL rewards.

Notes:
· We are aware of the Prisma Hack that happened in March, 2024, but it was located in a Periphery contract we don't use.
· Prisma Audits by Zellic, Nomoi & MixBytes.

- Halborn audit finished at 6th of November, it's the mix of 2 audits, the first lasted 5 weeks by a junior Halborn auditor, and the second, 1 month, by Senior Halborn SR:
  Beraborrow_Core_Re-Assessment_Smart_Contract_Security_Assessment_Re…

Previous audits we endured:
https://beraborrow.gitbook.io/docs/audits/audits
We'll soon add the Sherlock private competition, which ends the 7th of February.

The stablecoin that gets minted as debt against the collateral tokens is named Nectar ($NECT), and features flash-loans and EIP-2612.

sNECT stands for staked NECT, it's the share token of the LiquidStabilityPool (LSP).

What are the mempool properties and blocktimes of Berachain?
You can find all info here: https://docs.berachain.com/learn/what-is-beaconkit

Which oracles do we plan on using?
Short term Redstone Oracle (push) and Chronicle, medium term Chainlink (if they deploy).

Is there a list of all collaterals?
In the tests we use the following collaterals: `bHoney`, `iBGT` and `honeyUsdcKodiakIsland`.

PumpBTC for BoycoVault and USDC for PSMBond.

Additional Links:

Documentation: https://beraborrow.gitbook.io/docs
Website: https://www.beraborrow.com/
Twitter: https://x.com/beraborrow
Discord: https://discord.com/invite/beraborrowofficial

On what chains are the smart contracts going to be deployed?

Berachain mainnet
If you are integrating tokens, are you allowing only whitelisted tokens to work with the codebase or any complying with the standard? Are they assumed to have certain properties, e.g. be non-reentrant? Are there any types of weird tokens you want to integrate?

We only allow whitelisted assets in the system.

The Tokens that we have integrated with have (up to) the following properties:

ERC20Upgradeable

ERC4626Upgradeable

OwnableUpgradeable

UUPSUpgradeable

Ownable

In addition to the above, we will be integrating a number of BTC and ETH based derivatives that will be bridged to Berachain, these are NOT supposed to be rebasing tokens.

uniBTC

pumpBTC

sBTC

eBTC

solvBTC

LBTC

ylBTCLST

BeraETH

WETH

STONE

ylstETH

USDC

USDe

yIfBTC

WBTC-HONEY Kodiak Island
WETH-HONEY Kodiak Island
WBTC-WETH Kodiak Island

Are there any limitations on values set by admins (or other roles) in the codebase, including restrictions on array lengths?

We do impose certain limits on what the owner (and other roles) can set:
DenManager: MCR must be at least 110% in most cases, and can't exceed the CCR defined in the core.
LiquidStabilityPool: Tokens can't already be added or locked, thresholds can't exceed 100% in basis points, and we require active price feeds.
LiquidationManager: The fee split must add up to exactly 100%.
PriceFeed: Heartbeats can't go past 2 days, and zero addresses or unconfigured feeds aren't allowed.
BeraborrowCore: Various fees (like NECT fee, denManager fee, etc.) can't go over 100%. Ownable 2 step.
BorrowerOperations: Minimum net debt has to be > 0, and any changes can't break MCR or CCR requirements.
DebtToken: Accepts multiple protocol instances doing authorized calls like mint, burn, or flash-mint. Flash-mints can theoretically be done up to type(uint).max - totalSupply().

Hardcoded variables:
BoycoVault Dens will be at a 300% ICR on opening.
- minNetDebt will be in a range between 100-1000 NECT, depending on expected Berachain gas prices (Liquity and Prisma have 2000).
- gasCompensation will be an order of magnitude less than minNetDebt
- redemptionFeeFloor and borrowingFeeFloor will be slightly greater than its respective price feed threshold
- maxRedemptionFee and maxBorrowingFee around 5%
- brimeMCR will be around 105%, since it's main use case is to absorb collateral redemptions, instead of them going to users and bootstrap NECT liquidity with no interest accrual neither fees.
- The liquidation coll gas compensation is 0.5% of the liquidated collateral, redistributed to various parties.
- For intra-portfolio rebalances, CollVaults and LiquidStabilityPool have swap slippage thresholds requirements.
- Entry and exit lsp fee can be set lower to rebalancer partners.
- CollVaults and LiquidStabilityPool (LSP) will launch with an UUPS proxy.
- LSP will have multiple protocol instances support (at least 2), whitelisting its through updateProtocol()`.

A full list of collateral assets that Beraborrow will integrate can be found in the following document:

https://docs.google.com/document/d/1o-__D_EcXpipBHLZq0FrVjt-NruHBoWmbD3AAqCQ_eo/edit

A list of our integrations and their documentation:
Chronicle and Redstone Oracle (Push) - They are our main price feeds
https://docs.chroniclelabs.org
https://docs.redstone.finance/docs/get-started/price-feeds/
https://bartio.beratrail.io/address/0x9B6f9f5DAF9906bFE0c6561B96F27326A269Fe12/contract/80084/code

OogaBooga - Native Berachain DEX aggregator, it's embedded in some LSPRouter and CollVaultRouter
https://docs.oogabooga.io/developers/swap-api
https://bartio.beratrail.io/address/0xF6eDCa3C79b4A3DFA82418e278a81604083b999D/contract/80084/code

InfraredVault - PoL infrastructure abstraction, InfraredCollVaults deposit their main asset in these yield bearing vaults
https://infrared.finance/docs
https://infrared-dao.github.io/infrared-contracts/src/core/InfraredVault.sol/contract.InfraredVault.html

Kodiak Islands - Our kodiak islands spot price feed, Kodiak is a UniswapV3 fork, Islands is a vaulted LP position.
Similar to Arrakis V1 vault
https://github.com/ArrakisFinance/vault-v1-core/blob/main/contracts/ArrakisVaultV1.sol
https://bartio.beratrail.io/address/0x740A0a5c94A8EECc614999bE6A3ccAc13910c9b8/contract/80084/code

Is the codebase expected to comply with any specific EIPs?

EIP4626 -> vault token for CollVaults, BoycoVault and sNECT (yield bearing stablecoin)
EIP712 -> LiquidationManager, Pollen and Nectar tokens.

Permits in LM are only implemented to prove validator partners are running the liquidation infrastructure.

Are there any off-chain mechanisms involved in the protocol (e.g., keeper bots, arbitrage bots, etc.)? We assume these mechanisms will not misbehave, delay, or go offline unless otherwise specified.

Yes, there are several off-chain components:

- **Oracle updaters** (Keeper bots):
Since Chronicle doesn't has roundId system, we use keepers to replicate the data structure to be compatible with PriceFeed.sol `getRoundData()`.
- **Liquidators**:
We partner with certain addresses (liquidation bots) to run the liquidation process. They receive a portion of the liquidation rewards. They must provide a “permit” signature verifying their role as an official or partner liquidator.
Rebalancing (in CollVaults and LSP):
Off-chain calls can be made by the owner or designated rebalancers to rebalance collateral within a specified slippage threshold.

- **BoycoVault emergency debt repayment / collateral recovery**

We assume these off-chain mechanisms do not maliciously censor or withhold updates. If they do misbehave or go offline, the protocol may suffer from stale prices, delayed liquidations, or suboptimal rebalances. But general health improvements are appreciated.

What properties/invariants do you want to hold even if breaking them has a low/unknown impact?

Vault share balance consistency:
In each vault (e.g., LiquidStabilityPool, InfraredCollVault), totalAssets() should align with the underlying holdings. We do not allow any deposit or donation to artificially change share prices.
No Arb loop anywhere.
EOV and MEV can be limited with current fees.
Multiple protocol instances are compatible (more info in below section).

Please discuss any design choices you made.

Multi-collateral adaptation:
We forked Prisma (which itself forks Liquity) to allow multiple collateral types. We did so by introducing CollVault wrappers for each collateral.
Fees:
We have more flexible fee parameters (mint, redemption, flash-loan fees). Because they can be changed by governance, we do not overcomplicate the code with hard-coded invariants.
Flash loans:

We explicitly allow flash loans of NECT. This might invite advanced strategies or arbitrage, but we carefully virtually track users debt and collateral to ensure the system remains solvent.

Ignoring some micro-optimizations:

We do not accumulate donations in a way that changes share prices (to avoid complexity). Instead, donations are recognized for the protocol owner to claim or to deposit as vested incentives.

Focus on upgradeability:

Certain contracts (like the LSP and CollVaults) are upgradeable through a proxy. This is to ensure future expansions or fixes, especially given the novel yield aggregator logic behind sNECT and Infrared depositing.

LiquidStabilityPool:

Prior CDP based protocols such as Liquity, Prisma etc (from which we are based on) use a stability pool as a buffer against liquidations. We have tokenised this stability pool as an ERC-4626.

MetaBeraborrowCore, LSP and DebtToken have been developed/modified to support multiple protocol instances, with NECT and sNECT (LSP) being the same contract. This is mainly done to avoid `getTCR()` to reach values close to block size limit (with 20 InfraredCollateralVaults per protocol, 1/6th of the Berachain block gas limit (30M) is reachable by a simple LSP::deposit or BorrowerOperations::OpenDen).

We plan to launch with a protocol instance with only PermissionedDenManagers linked to it's BoycoVault (you have a photo in the bottom of this document of the assets we'll support in this protocol).

And then progressively launch normal public DenManagers in another protocol.

Major contract changes

DebtToken (NECT) ->

We made protocol addresses no longer immutable and to be changed to support another protocol having permissioned control over NECT since we may want to deploy another protocol that also can mint and burn NECT.

- We will want to support the Liquity V2 based fork and the current implementation. NECT must be mintable by both of these protocols. We don’t want a situation where NECT is only mintable by the current implementation.

- NECT can be flash loaned. Be aware of flash loan-related balance manipulations or arb loops.

DenManager -> The collateral is a multi-asset vaulted version, to earn PoL rewards. We have deleted POLLEN rewards.
LiquidStabilityPool (sNECT) -> Almost everything was changed, it consists of an upgradeable ERC4626 by the **UpgradeableProxy**, with fees taken in shares (sNECT). Deposits/mints are taken in NECT, and withdraws/redeem can either:

- Give proportional NECT/liquidated collateral/extra assets such as $POLLEN.
- Or, select the order of the tokens you want to withdraw, which are passed in the **preferredUnderlyingTokens** array parameter. NECT is enforced to be the last token.

Liquidated collateral and 'extraAssets' balance updates are progressively unlocked based on the `src/libraries/EmissionsLib` library.
Owner can rebalance the underlying portfolio assets with the condition that the end result doesn't have a slippage higher than a certain governance controlled threshold.
Vault doesn't account for donations, and these are receivable by the owner via `receiveDonations()`.
TLDR;

- Accrues fees: minting, redemption.
- Supports both collateral and underlying assets.
- Only accepts deposits in NECT.
- Pro-rata redeems all underlying assets.
- Order based redeems of underlying assets.
- Donations (ERC-20 transfer) don't add extra value to the vault (ratio of totalAssets()/totalSupply() stays the same)
- Reward emissions. Rewards are unvested over time. Rewards must be protected from yield front-running.
- Owner can rebalance within the allowed slippage threshold.

PriceFeed -> This implementation used to cache price records, but we have removed it to have staticallable `fetchPrice()`.

- Implementation ported from 'external' to 'external view'
- Works with both Chronicle through an adapter, Redstone and Chainlink 'backends'
- Redirects calls to custom pricing oracle for CollVaults and other assets which are derivatives, hence can be calculated in spot (e.g. bHoney and Kodiak LPs Shares)

InfraredCollVault. Abstract contract for all CollVaults which earn PoL (Proof of Liquidity) rewards by staking underlying collateral.

- Similar to LiquidStabilityPool: rebalancing, donations, pricing, fees etc.
- Asset is wrapped into vault share token, which it's put into the den.
- iRED emissions are not harvested. They're escrowed on the contract, considered as a donation, and kept for later.
- totalAssets(), preview() functions account for "future rewards" from iBGT Vault
- Has spot pricing of share token (Remultiplies Asset price by totalAssets()/totalSupply() of Vault.
- PriceFeed.sol handles it differently through looking at `whitelistCollateralVault`.

BaseCollVault. Abstract contract used by **InfraredCollVault**, it should have the property of being upgradeable to an **InfraredCollVault.**

iBGTVault. Infrared iBGT Vault wrapper. Generates yield by staking iBGT into Infrared.

- Inherits InfraredCollateralVault.

bHoneyVault. Present in a few tests, but won't be used in mainnet.

KodiakIslandsVault. Kodiak Islands Share Vault Wrapper. Also generates yield by staking the Share tokens into Infrared.

- Inherits InfraredCollateralVault.
- Kodiak Islands are a vaulted representation of a concentrated liquidity LP position into Kodiak, a Uniswap V3 fork.
- The earned iBGT is compounded by depositing in **ours** previously mentioned **iBGTVault**.
- The pricing solution gets the underlying reserves of the Island position and multiplies the tokens amounts for its respective price.
- Pricing implementation on spotOracles/**KodiakIslandFeed.sol** is based on this implementation given by the Kodiak team, but modified to be agnostic of token0 and token1 decimals.

ChronicleWrapper. Wraps a ChronicleOracle to have a Chainlink interface to be integrated to PriceFeeds.sol.

- RoundIds logic is not present in Chronicle, and PriceFeed.sol queries the previous response to compare for max deviation threshold.
  For some assets we will need to create ChronicleWrapper instances, and a keeper like Gelato or Chainlink will `storeNewPrice` every time it detects a new published price update, since PriceFeed.sol checks for maxDeviation.
- The price validation happens at the PriceFeed.sol lovel.

CollVaultRouter. Abstracts away interacting with CollVaults, enabling users to deposit the raw collateral, it has slippage checks and custom `_preDeposit` hooks logic.

BrimeDen. Allows the protocol to mint NECT more efficiently in order to boost NECT liquidity.

- Only protocol owner can interact with BrimeDen
- Zero minting fees
- Zero interest (we acknowledge that increases the real interest others have to pay, we will counter-effect this with lowering gross interests when the brimeDen debt over the total system debt increases)
- Has redemption fees (to make sure the the system is stable)
- Lower MCR to absorb collateral redemptions, instead of them going to users.

LiquidationManager.
The DEBT_GAS_COMPENSATION and collGasCompensation are split between the **ValidatorPool**, sNECT gauge (PoL infrastructure where the LSP share token is incentivized) and the liquidator that initiated the liquidation.
LiquidaitonManager has been extended in two ways:

- Fee sharing. Instead of just rewarding liquidators, Beraborrow rewards 3 actors:
    - Liquidator. Same as in the original protocol.
    - sNECT Gauge. Part of the rewards are distributed into PoL sNECT gauge.
    - Fee Pool. List of our partner liquidators, participating in liquidations and PoL incentives.
- Permits. Protocol relies on partner liquidators to be available and run their liquidator bot software. In order to prove that they're running a liquidation bot software, they would provide a permit signature for liquidations. Permits here act as attestations.

ValidatorPool. Distributes liquidated collateral rewards and NECT to an amount of whitelisted validators, each with different shares that sum to BP (1e4).

LSPRouter. Abstracts the vaulted assets to the end user, interfaces pro-rata redeems, order redeems, and offers swap abstractions on deposits and withdrawals with OogaBooga.

LSPGetters. ExtSload based approach to avoid bytecode size limits on LSP implementation, uses LSPStorageLib to get slots.

Boyco

Note: List of assets we'll support in Boyco

Beraborrow	
	uniBTC
	pumpBTC
	sBTC
	eBTC
	solvBTC
	LBTC
	ylBTCLST
	BeraETH
	wETH
	STONE
	ylstETH
	USDC
	USDe
	ylfBTC
	wBTC-HONEY (Kodiak Island)
	wETH-HONEY (Kodiak Island)
	wBTC-wETH (Kodiak Island)
	solvBTC.bbn

1) **PSMBond** contract.
Accepts a stablecoin (e.g. USDC), mints NECT (our protocol stablecoin) 1:1 in exchange (e.g. 20 USDC deposit with mint 20 NECT). Users end up with NECT, our protocol ends up with USDC. USDC is later used in various ways, including:
- acting as an 'insurance fund' for our stablecoin
- supplied as a collateral to mint NECT

Input = USDC, output = NECT

onlyBoyco modifier only withelists a Weiroll Wallet, that will immutably execute the following script:
https://github.com/EnsoFinance/shortcuts-registry/blob/main/src/shortcuts/beraborrow/nect-honey.ts
https://docs.royco.org/more/cross-chain-deposit-module-ccdm/deposit-executor#executing-deposit-recipes

2) **BoycoVault** contract.
BoycoVault is a tokenized position with the set CR. It accepts various volatile tokens (mostly different BTC and ETH liquid staking derivatives), and later supplies them as a collateral with the set CR and mints NECT. For example, given pumpBTC is priced at $95000 per token and BoycoVault has 300% CR, when a user supplies 0.1 pumpBTC, the protocol mints 0.1 * $95000 * 100% / 300% = 3166 NECT. Minted NECT is them deposited into Liquid Stability Pool (tokenized version of Liquity's Stability Pool). BoycoVault is ERC4626 that mints shares that represent proportional ownership of the position.
Input = volatile tokens, output = 4626 vault shares representing shared ownership of the position.

.env file:

```
OWNER=0x2347F750c67Eb6741D4be331861eE8DF9137Cc40
GUARDIAN=0xb39E9Cf7bF352dE55390eF0D898D81E18B3FC309
FEE_RECEIVER=0x2347F750c67Eb6741D4be331861eE8DF9137Cc40
LAYER_ZERO_ENDPOINT=0xb39E9Cf7bF352dE55390eF0D898D81E18B3FC312
MANAGER=0x2347F750c67Eb6741D4be331861eE8DF9137Cc40
LOCK_TO_TOKEN_RATIO=1000000000000000000
WBERA=0x7507c1dc16935B82698e4C63f2746A2fCf994dF8
NECTAR=0xf5AFCF50006944d17226978e594D4D25f4f92B40
IBGT=0x46eFC86F0D7455F135CC9df501673739d513E982
HONEY=0x0E4aaF1351de4c0264C5c7056Ef3777b41BD8e03
BHONEY=0x1306D3c36eC7E38dd2c128fBe3097C2C2449af64
IRED=0xE9eEa54fB348b8B4A350FE88ae8DB6E1A7a39Ae0
POLLEN=0xa591eef221369321De76d958dC023936Fb39B26A
NECTAR=0xf5AFCF50006944d17226978e594D4D25f4f92B40
USDC=0xd6D83aF58a19Cd14eF3CF6fe848C9A4d21e5727c
KODIAK_ISLAND_HONEY_USDC=0xb73deE52F38539bA854979eab6342A60dD4C8c03
IBGT_INFRARED_VAULT=0x31E6458C83C4184A23c761fDAffb61941665E012
BERA_FEED=0xd5F3478253957777fAb22D4Ed216dcC750b6b141
NECT_FEED=0x2F42cA5722583651a07684a79350Fb1a5f932fBf
IBGT_FEED=0xCaA6114499B7615CB37c605CCdC3Caf96745334B
POLLEN_FEED=0x61C3f99E4d0Bdf31f3D2965AfbE3a0A2cEc21793
HONEY_FEED=0xc01382f6Cc33ddB499fB672F762E8BDCAF3E8f8E
USDC_FEED=0x3813e353a55dae4cd5fc2d21e2197670a2394de5
BHONEY_INFRARED_VAULT=0x7d91bf5851b3a8bcf8c39a69af2f0f98a4e2202a
KODIAK_ISLAND_HONEY_USDC_INFRARED_VAULT=0x1B602728805Ca854e0DFDbbBA9060345fB26bc20
```

```
OB_ROUTER=0xF6eDCa3C79b4A3DFA82418e278a81604083b999D
SNECT_GAUGE=0x72e222116fC6063f4eE5cA90A6C59916AAD8352a
ENTRY_FEE_BP=100
EXIT_FEE_BP=120
CCR=1500000000000000000
LIQUIDATOR_COMPENSATION_FEE=100000000000000000 # 1e17
SNECTGAUGE_COMPENSATION_FEE=800000000000000000 # 8e17
POOL_COMPENSATION_FEE=100000000000000000 # 1e17
LSP_BOOTSTRAP_PERIOD=0
DM_BOOTSTRAP_PERIOD=0

RPC_URL=http://localhost:8545
CARTIO_WEIROLL=0x5c6Ee304399DBdB9C8Ef030aB642B10820DB8F56
PUMP_BTC=0xE1F167CDE04d5d0F8d096957b3A23a7005618976
OB_API_KEY=
# Anvil localhost RPC URL
RPC_URL=http://localhost:8545
```