B-Harvest Kava

Kava CDP/Auction Audit Report

By B-Harvest

2020-05-20

Audit Summary 2

Scope of Work 2

- Security Audit 2
- Core CDP Functionality: 2
- Core Auctions Functionality: 3
- Golang review 3
- Financial/Scenario Audit 3

Limitations and Assumptions 4

- Security Audit 4
- Financial/Scenario Audit 4

Security Audit 5

- Core CDP Functionality 5
- Core Auctions Functionality 7
- Golang review 7

Financial / Scenario Audit 17

- External Oracle Feeds and Market Volatility 17
- Kava CDP Insolvency - Debt Auction 18
- Kava CDP Insolvency - CDP/Auction Parameters Review 20
- USDX Price Stability Review 22
- Review of “Black Thursday” in MakerDAO 23

Conclusion 25

References 26

1. Audit Summary

In security and technical review section, it was verified with a focus whether the implementation performs state transitions that are detailed in the specification. Furthermore, we audited safe error, panic processing according to the characteristics of the on-chain logic. And from golang point of view, we also verified general errors and performances of code.

In financial/scenario review section, we analyzed different kinds of risks which are related to several environment factors such as unstable oracle feeds, high short-term market volatilities, economic design of debt auction at CDP insolvency, global parameter structures and stable token mechanism.

In addition, thanks to the recent Black Thursday event which caused severe functional failure of MakerDAO network, we could add additional risk analysis based on the event and its outcomes.

2. Scope of Work

Target Repository : `https://github.com/Kava-Labs/kava`
Target Commit Hash : `8f3858509a0aff6ed26767d35c6ea5f64d808e03`
Duration of Work : 2020-05-06 ~ 2020-05-20

Security Audit

Core CDP Functionality:

Compare to spec in `x/cdp/spec`

To implementation, specifically in:

- cdp/types
  `x/cdp/types`
- cdp/keeper
  `x/cdp/keeper`
- cdp - begin blocker
  `x/cdp/abci.go`

Core Auctions Functionality:

Compare to spec in `x/auction/spec`

To implementation, specifically in:

- auction/types
  `x/auction/types`
- auction/keeper
  `x/auction/keeper`
- auction - begin blocker
  `x/auction/abci.go`

Verification that the implementation performs all state transitions that are detailed in the specification. Any inconsistencies must be pointed out.

Verifying that the implementation does not perform any state transitions that are not detailed in the specification.

Depending on the selected issues and findings, deliver a document which details:

1. Details of the Findings
2. Suggestions / Comments

Golang review

- Enforce that the golang files are written against best practices to minimize the chance of a panic while the application is running
- General performance of code
- Issues and comments dependent on cosmos-sdk, tendermint

Financial/Scenario Audit

Depending on the selected scenario, deliver a document which details:

1. Background
2. Defining risks
3. Scenario Analysis
4. Suggested Adjustment
5. Any applicable appendix

3. Limitations and Assumptions

Security Audit

The CDP module relies on an external pricefeed. For purposes of this audit, assume that the pricefeed is well-functioning (reporting prices in a timely manner) and honest (reporting the accurate price of an asset).

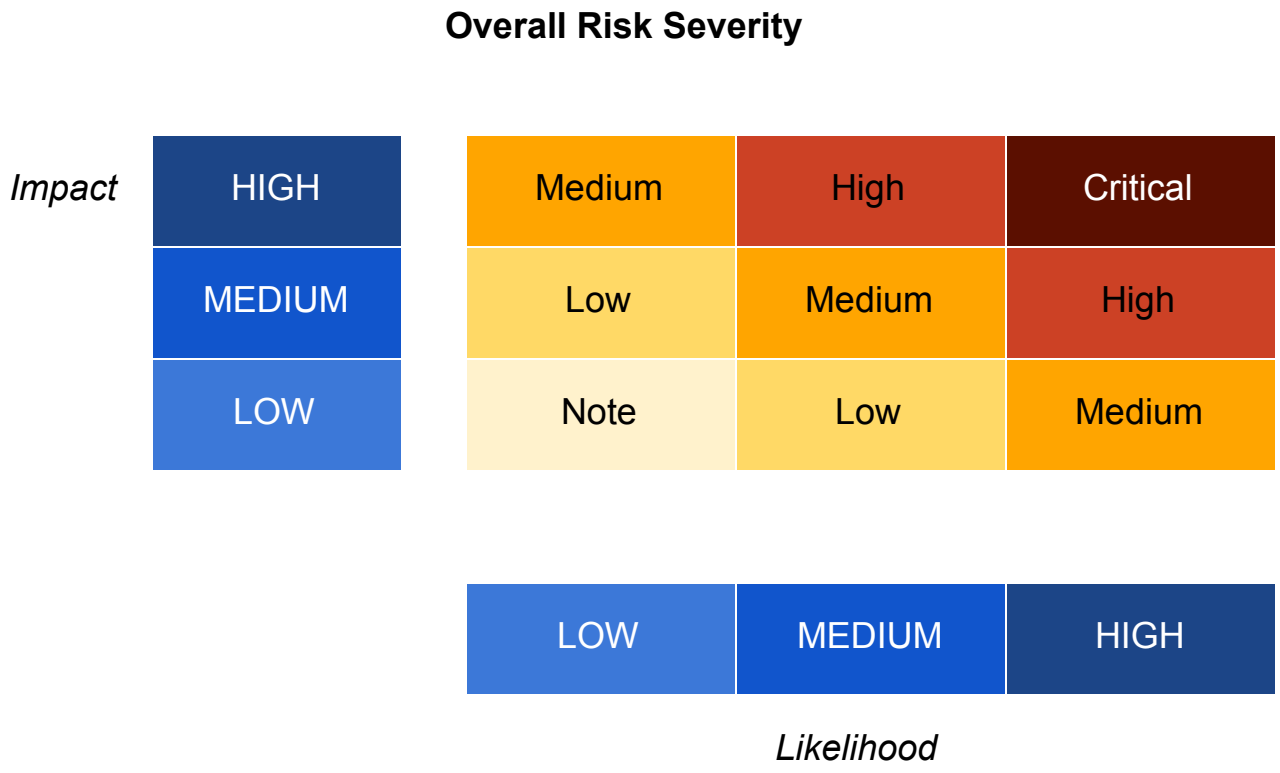
The CDP and Auction modules' parameters are governed by the token holders of the Kava chain. For purposes of this audit, assume that parameters are chosen in such a way that does not put the custody of CDP holder's collateral at risk, nor impacts the overall functioning of the state machine. Namely, assume that an outside observer or system participant will (1.) have sufficient time to respond to any changes in parameters and (2.) chosen parameter values will not dramatically increase the size of blockchain state (e.g. setting the auction size parameter to 1 => then any liquidation would trigger millions of auctions and slow the chain to a halt).

Financial/Scenario Audit

The Financial/Scenario cannot cover the pricefeed related risk because the module is out of scope for this audit. This report assumes that Kava Protocol will directly accept oracle values from external oracle feeds.

4. Security Audit

Severity calculated according to the OWASP Risk Rating Methodology based on Impact and Likelihood



4.1. Core CDP Functionality

Needed invariant checking logic [*Note*]

Details of the Findings with Suggestions / Comments

- Add `x/cdp/keeper/invariants.go` file to check `totalPrincipal`, `totalDebt`, Collateral, fee amount, cdp index(by owner, by denom, by `CollateralRatio`) like same level as auction module

Differences between codes and specs, missing spec documents [*Note*]

Details of the Findings with Suggestions / Comments

- Missing `SavingsRateMacc` in `x/cdp/spec/02_state.md`

```
 ## Module Accounts

-The cdp module account controls two module accounts:
+The cdp module account controls three module accounts:

 **CDP Account:** Stores the deposited cdp collateral, and the debt coins for the debt in all
the cdps.

 **Liquidator Account:** Stores debt coins that have been seized by the system, and any
stable asset that has been raised through auctions.

+**SavingRate Account:** SavingsRateMacc module account for savings rate
+
 ## CDP
```

- Missing `SurplusThreshold, DebtThreshold` in `x/cdp/spec/06_params.md`

```
| SavingsDistributionFrequency | string (int)            | "84600"
| number of seconds between distribution of the savings rate       |
+| SurplusThreshold             | string (int)            | "1000000000"
| TBD                                                              |
+| DebtThreshold                | string (int)            | "1000000000"
| TBD                                                              |
| CircuitBreaker               | bool                    | false
| flag to disable user interactions with the system                |
```

- Update Example of Parameter `ReferenceAsset` in `x/cdp/spec/06_params.md`

```
 | Denom            | string       | "usdx"     | pegged asset coin denom
|
-| ReferenceAsset   | string       | "USD"      | asset this asset is pegged to,
informational purposes only                                          |
+| ReferenceAsset   | string       | "usd"      | asset this asset is pegged to,
informational purposes only                                          |
 | ConversionFactor | string (int) | "6"        | 10^_ multiplier to go from external amount
(say $1.50) to internal representation of that amount (1500000) |
```

4.2. Core Auctions Functionality

Differences between codes and specs, missing spec documents [*Note*]

Details of the Findings with Suggestions / Comments

- `Surplus Reverse Auction` ➔ `Collateral Auction` in `x/auction/spec/01_concepts.md`

```
  * **Surplus Auction:** An auction in which a ...
 * **Debt Auction:** An auction in which a ...
-* **Surplus Reverse** Auction:** Are two phase auction is which a ...
+* **Collateral** Auction:** Are two phase auction is which a ...
```

4.3. Golang review

To make panic instead of return error for state safety [*Low*]

If a change and kv set in state has already occurred in a unit of function, it is safe to create a panic to avoid an incomplete change in state when an unexpected error occurs.

There could have already been a change and set in status on the preceding code in the function

- `x/cdp/keeper/cdp.go` #L94-L96

```
    // update total principal for input collateral type
    k.IncrementTotalPrincipal(ctx, collateral.Denom, principal)

    // set the cdp, deposit, and indexes in the store
    collateralToDebtRatio := k.CalculateCollateralToDebtRatio(ctx, collateral, principal)
    err = k.SetCdpAndCollateralRatioIndex(ctx, cdp, collateralToDebtRatio)
    if err != nil {
-       return err
+       panic(err)
    }
    k.IndexCdpByOwner(ctx, cdp)
```

There are similar cases in

- `x/auction/keeper/auctions.go` #L30, #L68
- `x/auction/keeper/keeper.go` #L85
- `x/cdp/keeper/auctions.go` #L91, #L107, #L183, #L189, #L221, #L230
- `x/cdp/keeper/deposit.go` #L51, #L101
- `x/cdp/keeper/draw.go` #L73, #L146, #L165

- x/cdp/keeper/seize.go #L64

There could have already been a change and set in status on iteration

- x/cdp/keeper/seize.go #L85, and similarly in x/cdp/keeper/fees.go #L90

```
        for _, c := range cdpsToLiquidate {
                err := k.SeizeCollateral(ctx, c)
                if err != nil {
-                       return err
+                       panic(err)
                }
        }
        return nil
```

Format string bug [*Low*]

Details of the Findings

- In x/bep3/types/asset.go #L36, `func (a AssetSupply) String()`, return misconfigured formatstring

Suggestions / Comments

```
        "\n    Outgoing supply:    %s"+
        "\n    Current supply:     %s"+
-       "\n    Limit:       %s"+
+       "\n    Limit:       %s",
        a.Denom, a.IncomingSupply, a.OutgoingSupply, a.CurrentSupply, a.Limit)
```

Difference between function name and actual behavior [*Low*]

Details of the Findings

- Function name `SignedPercetageIsOverThreshold` and comments used over, expended, but code works as `GTE` in x/validator-vesting/types/validator_vesting_account.go #L51-L56

Suggestions / Comments

- Unify code action with function name, comments

Code quality, styling optimization [*Note*]

`sdk.Int`, `sdk.Dec` positive/negative discriminant optimization

- x/auction/simulation/operations.go #L136, #L183

```
            // Check auction can still receive new bids
```

```
-               if a.Lot.Amount.Equal(sdk.ZeroInt()) {
+               if a.Lot.Amount.IsZero() {
                        return sdk.Coin{}, errorCantReceiveBids
                }
```

```
                // Check auction can still receive new bids
-               if a.IsReversePhase() && a.Lot.Amount.Equal(sdk.ZeroInt()) {
+               if a.IsReversePhase() && a.Lot.Amount.IsZero() {
                        return sdk.Coin{}, errorCantReceiveBids
                }
```

- x/cdp/keeper/auctions.go #L51, #L104

```
        totalCollateral := deposits.SumCollateral()
-       for totalCollateral.GT(sdk.ZeroInt()) {
+       for totalCollateral.IsPositive() {
                for i, dep := range deposits {
```

```
        }
-       if partialAuctionDeposits.SumCollateral().GT(sdk.ZeroInt()) {
+       if partialAuctionDeposits.SumCollateral().IsPositive() {
```

- x/cdp/keeper/draw.go #L180

```
        proposedBalance := cdp.Principal.Amount.Sub(payment.Amount)
-       if proposedBalance.GT(sdk.ZeroInt()) && proposedBalance.LT(dp.DebtFloor) {
+       if proposedBalance.IsPositive() && proposedBalance.LT(dp.DebtFloor) {
```

- x/cdp/types/params.go #L306, #L329

```
-               if cp.LiquidationPenalty.LT(sdk.ZeroDec()) ||
cp.LiquidationPenalty.GT(sdk.OneDec()) {
+               if cp.LiquidationPenalty.IsNegative() ||
cp.LiquidationPenalty.GT(sdk.OneDec()) {
```

```
-       if debtParam.SavingsRate.LT(sdk.ZeroDec()) || debtParam.SavingsRate.GT(sdk.OneDec())
{
+       if debtParam.SavingsRate.IsNegative() || debtParam.SavingsRate.GT(sdk.OneDec()) {
```

- x/cdp/types/utils.go #L78

```
        z = x
-       if n.Mod(sdk.NewInt(2)).Equal(sdk.ZeroInt()) {
+       if n.Mod(sdk.NewInt(2)).IsZero() {
                z = b
```

```
        }

        halfOfB := b.Quo(sdk.NewInt(2))
        n = n.Quo(sdk.NewInt(2))

-       for n.GT(sdk.ZeroInt()) {
+       for n.IsPositive() {
                xSquared := x.Mul(x)
```

Unnecessary typecasting

- x/auction/types/keys.go
- x/bep3/types/keys.go
- x/committee/types/keys.go

```
 func Uint64ToBytes(id uint64) []byte {
        bz := make([]byte, 8)
-       binary.BigEndian.PutUint64(bz, uint64(id))
+       binary.BigEndian.PutUint64(bz, id)
        return bz
 }
```

Unused Arguments

- In x/cdp/keeper/auctions.go #L144, function CreateAuctionFromPartialDeposits, unused arguments debt and collateral, It could remove and fix usages

Duplicated parameter types

- x/cdp/types/cdp.go #L22

```
 // NewCDP creates a new CDP object
-func NewCDP(id uint64, owner sdk.AccAddress, collateral sdk.Coin, principal sdk.Coin, time time.Time) CDP {
+func NewCDP(id uint64, owner sdk.AccAddress, collateral, principal sdk.Coin, time time.Time) CDP {
```

- x/cdp/keeper/cdp.go

```
 // AddCdp adds a cdp for a specific owner and collateral type
-func (k Keeper) AddCdp(ctx sdk.Context, owner sdk.AccAddress, collateral sdk.Coin, principal sdk.Coin) error {
+func (k Keeper) AddCdp(ctx sdk.Context, owner sdk.AccAddress, collateral, principal sdk.Coin) error {
```

```
-func (k Keeper) MintDebtCoins(ctx sdk.Context, moduleAccount string, denom string, principalCoins sdk.Coin) error {
+func (k Keeper) MintDebtCoins(ctx sdk.Context, moduleAccount, denom string, principalCoins sdk.Coin) error {
```

```
-func (k Keeper) BurnDebtCoins(ctx sdk.Context, moduleAccount string, denom string, paymentCoins sdk.Coin) error {
+func (k Keeper) BurnDebtCoins(ctx sdk.Context, moduleAccount, denom string, paymentCoins sdk.Coin) error {
```

```
-func (k Keeper) ValidateCollateralizationRatio(ctx sdk.Context, collateral sdk.Coin, principal sdk.Coin, fees
sdk.Coin) error {
+func (k Keeper) ValidateCollateralizationRatio(ctx sdk.Context, collateral, principal, fees sdk.Coin) error {
```

```
-func (k Keeper) CalculateCollateralToDebtRatio(ctx sdk.Context, collateral sdk.Coin, debt sdk.Coin) sdk.Dec {
+func (k Keeper) CalculateCollateralToDebtRatio(ctx sdk.Context, collateral, debt sdk.Coin) sdk.Dec {
```

```
-func (k Keeper) CalculateCollateralizationRatio(ctx sdk.Context, collateral sdk.Coin, principal sdk.Coin, fees
sdk.Coin) (sdk.Dec, error) {
+func (k Keeper) CalculateCollateralizationRatio(ctx sdk.Context, collateral, principal, fees sdk.Coin) (sdk.Dec,
error) {
```

- x/auction/keeper/auctions.go #L46

```
-func (k Keeper) StartDebtAuction(ctx sdk.Context, buyer string, bid sdk.Coin, initialLot sdk.Coin, debt sdk.Coin)
(uint64, error) {
+func (k Keeper) StartDebtAuction(ctx sdk.Context, buyer string, bid, initialLot, debt sdk.Coin) (uint64, error) {
```

- x/auction/types/auctions.go #L165

```
-func NewDebtAuction(buyerModAccName string, bid sdk.Coin, initialLot sdk.Coin, endTime time.Time, debt sdk.Coin)
DebtAuction {
+func NewDebtAuction(buyerModAccName string, bid, initialLot sdk.Coin, endTime time.Time, debt sdk.Coin) DebtAuction {
```

- ...
- There are similar cases in `CreateAuctionsFromDeposit`, `CreateAuctionFromPartialDeposits`, `NewParams`, `RelativePow`, `NewCommitteeDeleteProposal`, `NewQueryAllAuctionParams`, `operationClaimAtomicSwap`, `NewQueryAtomicSwaps`, `NewAtomicSwap`, `DepositCollateral`, `WithdrawCollateral`, `ValidatePaymentCoins`, `calculatePayment`, `GetTotalPrincipal`, `SetTotalPrincipal`, `LiquidateCdps`, `NewMsgCreateCDP`, `NewMsgDeposit`, `NewMsgWithdraw`, `NewProposal`, `NewCommitteeChangeProposal`, `GetTotalPrincipal`, `NewReward`, `NewRewardPeriod`, `NewPeriod`, `CreateAtomicSwap`, `ClaimAtomicSwap`, `NewMsgCreateAtomicSwap`

Unused function, Replace with code of the same function

- In `x/bep3/types/keys.go` #L54-L58, function `BytesToHex` unused and same function with `hex.EncodeToString()`

Unused internal constants

- In `x/bep3/types/params.go` #L13, const `bech32MainPrefix` is unused

Unnecessary `fmt.Sprintf`, redundant format string

- x/cdp/types/genesis.go #L61-L65

```
        if err := sdk.ValidateDenom(gs.DebtDenom); err != nil {
-               return fmt.Errorf(fmt.Sprintf("debt denom invalid: %v", err))
+               return fmt.Errorf("debt denom invalid: %v", err)
        }

        if err := sdk.ValidateDenom(gs.GovDenom); err != nil {
-               return fmt.Errorf(fmt.Sprintf("gov denom invalid: %v", err))
+               return fmt.Errorf("gov denom invalid: %v", err)
        }
```

- x/bep3/types/params.go #L201-L217

```
    if asset.CoinID < 0 {
-       return fmt.Errorf(fmt.Sprintf("asset %s must be a non negative integer",
asset.Denom))
+       return fmt.Errorf("asset %s must be a non negative integer", asset.Denom)
    }

    if !asset.Limit.IsPositive() {
-       return fmt.Errorf(fmt.Sprintf("asset %s must have a positive supply limit",
asset.Denom))
+       return fmt.Errorf("asset %s must have a positive supply limit", asset.Denom)
    }

    _, found := coinDenoms[asset.Denom]
    if found {
-       return fmt.Errorf(fmt.Sprintf("asset %s cannot have duplicate denom", asset.Denom))
+       return fmt.Errorf("asset %s cannot have duplicate denom", asset.Denom)
    }

    coinDenoms[asset.Denom] = true

    _, found = coinIDs[asset.CoinID]
    if found {
-       return fmt.Errorf(fmt.Sprintf("asset %s cannot have duplicate coin id %d",
asset.Denom, asset.CoinID))
+       return fmt.Errorf("asset %s cannot have duplicate coin id %d", asset.Denom,
asset.CoinID)
    }
```

- x/bep3/types/swap.go #L62-L71

```
 func (a AtomicSwap) Validate() error {
    if len(a.Sender) != AddrByteCount {
-       return fmt.Errorf(fmt.Sprintf("the expected address length is %d, actual length is
%d", AddrByteCount, len(a.Sender)))
+       return fmt.Errorf("the expected address length is %d, actual length is %d",
AddrByteCount, len(a.Sender))
    }
    if len(a.Recipient) != AddrByteCount {
-       return fmt.Errorf(fmt.Sprintf("the expected address length is %d, actual length is
%d", AddrByteCount, len(a.Recipient)))
+       return fmt.Errorf("the expected address length is %d, actual length is %d",
AddrByteCount, len(a.Recipient))
    }
    if len(a.RandomNumberHash) != RandomNumberHashLength {
-       return fmt.Errorf(fmt.Sprintf("the length of random number hash should be %d",
RandomNumberHashLength))
```

```
+       return fmt.Errorf("the length of random number hash should be %d",
RandomNumberHashLength)
    }
    if !a.Amount.IsAllPositive() {
-       return fmt.Errorf(fmt.Sprintf("the swapped out coin must be positive"))
+       return fmt.Errorf("the swapped out coin must be positive")
    }
    return nil
 }
```

Unify indentation and style of `String()`

- In x/bep3/types/asset.go #L31-L36, Use `` `string string2` `` instead of "string" + "\n string2"

```
 // String implements stringer
 func (a AssetSupply) String() string {
-       return fmt.Sprintf("Asset Supply"+
-               "\n    Denom:              %s"+
-               "\n    Incoming supply:    %s"+
-               "\n    Outgoing supply:    %s"+
-               "\n    Current supply:     %s"+
-               "\n    Limit:       %s",
+       return fmt.Sprintf(`Asset Supply
+       Denom:              %s
+       Incoming supply:    %s
+       Outgoing supply:    %s
+       Current supply:     %s
+       Limit:       %s`,
                a.Denom, a.IncomingSupply, a.OutgoingSupply, a.CurrentSupply, a.Limit)
 }
```

- There are similar cases in `x/bep3/types/swap.go`, `x/incentive/types/rewards.go` and trivia `String()` indentation errors in `x/auction/types/auctions.go`, `x/auction/types/msg.go`, `x/cdp/types/cdp.go`, `x/cdp/types/deposit.go`, `x/cdp/types/msg.go`, `x/cdp/types/params.go`

Simplify for loop

- x/validator-vesting/types/validator_vesting_account.go #L89, #L122

```
        var vestingPeriodProgress []VestingProgress
-       for i := 0; i < len(periods); i++ {
+       for range periods {
                vestingPeriodProgress = append(vestingPeriodProgress, VestingProgress{false,
false})
        }
```

```
                VestingPeriods:     periods,
        }
        var vestingPeriodProgress []VestingProgress
```

```
-       for i := 0; i < len(periods); i++ {
+       for range periods {
                vestingPeriodProgress = append(vestingPeriodProgress, VestingProgress{false,
false})
        }
```

Typo in string literal

- In x/committee/types/msg.go #L9

```
-       TypeMsgSubmitProposal = "commmittee_submit_proposal" ...
+       TypeMsgSubmitProposal = "committee_submit_proposal" ...
```

Typo in method name

- x/validator-vesting/types/validator_vesting_account.go #L51-L52

```
-// SignedPercetageIsOverThreshold checks if the signed percentage exceeded the threshold
-func (cpp CurrentPeriodProgress) SignedPercetageIsOverThreshold(threshold int64) bool {
+// SignedPercentageIsOverThreshold checks if the signed percentage exceeded the threshold
+func (cpp CurrentPeriodProgress) SignedPercentageIsOverThreshold(threshold int64) bool {
        signedPercentage := cpp.GetSignedPercentage()
        return signedPercentage.GTE(sdk.NewDec(threshold))
 }
```

- x/validator-vesting/keeper/keeper.go #L121

```
        if sdk.NewDec(vv.CurrentPeriodProgress.TotalBlocks).IsZero() {
                successfulVest = true
        } else {
-               successfulVest =
vv.CurrentPeriodProgress.SignedPercetageIsOverThreshold(vv.SigningThreshold)
+               successfulVest =
vv.CurrentPeriodProgress.SignedPercentageIsOverThreshold(vv.SigningThreshold)
        }
```

Typos in comments and error msgs

- ≤ to < for .LT() in x/auction/keeper/auctions.go #L182

```
        if bid.Amount.LT(minNewBidAmt) {
-               return a, sdkerrors.Wrapf(types.ErrBidTooSmall, "%s ≤ %s%s", bid,
minNewBidAmt, a.Bid.Denom)
+               return a, sdkerrors.Wrapf(types.ErrBidTooSmall, "%s < %s%s", bid,
minNewBidAmt, a.Bid.Denom)
        }
```

- "nameservice" to "each module name" for comments about query in x/auction/client/cli/query.go #L18, also there are similar cases in module cdp, pricefeed, validator-vesting

```
 // GetQueryCmd returns the cli query commands for this module
 func GetQueryCmd(queryRoute string, cdc *codec.Codec) *cobra.Command {
-       // Group nameservice queries under a subcommand
+       // Group auction queries under a subcommand
        auctionQueryCmd := &cobra.Command{
                Use:   "auction",
                Short: "Querying commands for the auction module",
...
```

- Trivia typo in comments
  - rewardToDisribute -> rewardToDistribute
    - x/cdp/keeper/savings.go #L43
  - augmuented -> augmented
    - x/cdp/keeper/cdp.go #L443
  - poiint -> point
    - x/auction/types/genesis.go #L10
    - x/committee/types/genesis.go #L8
  - callled -> called
    - x/auction/keeper/keeper.go #L143

Reuse already initialized const variable

- In x/cdp/keeper/fees.go #L17, using BaseDigitFactor instead of 1000000000000000000

```
        feePerSecond := k.getFeeRate(ctx, denom)
-       scalar := sdk.NewInt(1000000000000000000)
+       scalar := sdk.NewInt(BaseDigitFactor)
```

Fix indentation go code in spec

- x/auction/spec/06_begin_block.md

```
 ```go
 var expiredAuctions []uint64
-       k.IterateAuctionsByTime(ctx, ctx.BlockTime(), func(id uint64) bool {
-               expiredAuctions = append(expiredAuctions, id)
-               return false
-       })
+k.IterateAuctionsByTime(ctx, ctx.BlockTime(), func(id uint64) bool {
+       expiredAuctions = append(expiredAuctions, id)
+       return false
+})

-       for _, id := range expiredAuctions {
-               err := k.CloseAuction(ctx, id)
-               if err != nil {
-                       panic(err)
-               }
```
```

```
-  }
+for _, id := range expiredAuctions {
+       err := k.CloseAuction(ctx, id)
+       if err != nil {
+              panic(err)
+       }
+}
 ```
```

Unify receiver names, `data` -> `gs` for `GenesisState`

- `x/committee/types/genesis.go` #L40-L48

```
-func (data GenesisState) Equal(data2 GenesisState) bool {
-       b1 := ModuleCdc.MustMarshalBinaryBare(data)
+func (gs GenesisState) Equal(data2 GenesisState) bool {
+       b1 := ModuleCdc.MustMarshalBinaryBare(gs)
        b2 := ModuleCdc.MustMarshalBinaryBare(data2)
        return bytes.Equal(b1, b2)
 }

 // IsEmpty returns true if a GenesisState is empty
-func (data GenesisState) IsEmpty() bool {
-       return data.Equal(GenesisState{})
+func (gs GenesisState) IsEmpty() bool {
+       return gs.Equal(GenesisState{})
 }
```

- There are similar cases `ExportGenesis` in `x/bep3/genesis.go`, `InitGenesis` in `x/validator-vesting/genesis.go`, `Equal`, `IsEmpty`, `ValidateGenesis` in `x/validator-vesting/types/genesis.go`
```

5. Financial / Scenario Audit

5.1. External Oracle Feeds and Market Volatility

Background

- Extreme oracle price moves are caused by
    - Oracle feed malfunctioning
    - Actual market price volatility
- Kava Protocol should be prepared necessary protection mechanisms to minimize risks caused by such extreme oracle price moves

Scenario Analysis

- Oracle price of a collateral asset temporarily hikes
    - Caused by failure or bug of oracle
    - Caused by failure or bug of exchange price API for oracle
    - Caused by extraordinary short term orderbook distortion by massive market orders in centralized exchange, intentional or unintentional
- Massive debt is raised under the very high asset prices, or maximum withdrawable collateral is withdrawn
- There exists very high risk that the debt will become under-collateral in near future when the oracle price normalized

Defining risks

- Oracle price from instant market order book → instability of short term oracle price
- The massive debt amount makes auction more difficult to meet the buyer with reasonable price in short time → concentrated debt creation

Suggested Adjustment

- Differentiate base price into two kinds
    - Base price for validating collateralization ratio
        - Collateralization ratio calculation
            - MIN(current price, median price over recent 30 minutes)

        - Reason : conservative collateralization ratio calculation to prevent too risky debt raising(or collateral withdraw) based on short term distorted oracle price
    - Base price for triggering liquidation of a debt
        - Collateralization ratio calculation
            - Median price over recent 30 minutes
        - Reason : use as recent price as possible not to delay liquidation, but avoid triggering massive amount of liquidation caused by very short term distorted oracle price
    - Why do we suggest using the median price over recent 30 minutes?
        - CME calculates most equity related future settlement price by 30 minutes VWAP(Volume Weighted Average Price)
        - In cryptocurrency market below problems create various manipulation risk on VWMA usage
            - Manipulated volume ➔ no volume weighted
            - Short term distorted price ➔ no average but median
        - To avoid utilizing manipulated volume and ignore outlier, we suggest using median price over recent 30 minutes
- Concentrated debt creation : diversification of debt creation
    - Time diversification : Debt creation ceiling for every day or week to limit too concentrated debt creation in short period of time
    - Liquidation price diversification : Debt creation ceiling for each liquidation price range so that the Kava Protocol does not experience too much debt liquidation at certain liquidation price

Any applicable appendix

- Nasdaq-100 futures settlement price calculation : https://www.cmegroup.com/confluence/display/EPICSANDBOX/Nasdaq-100

5.2. Kava CDP Insolvency - Debt Auction

Background

- Kava Protocol can face Kava CDP insolvency, result in minting new Kava and executing debt auction
- We review the process and strategy on resolving CDP insolvency and suggest alternative strategy in this section

Scenario Analysis

- When debt auction is executed, it is likely that most crypto assets are facing huge price drop in short time
- Minting kava at this kind of period will make kava auction price even more discounted because of kava supply increment in the market

Defining risks

- Market crash ➔ CDP insolvency ➔ Kava minted at discounted price ➔ amplified Kava dilution effect to all Kava investors ➔ resulting in higher price volatility of Kava in market crisis time
- The strategy that minting kava and resolving insolvency at the same time should be revisited

Suggested Adjustment

- General
    - Mint kava in normal times and auction it for USDX and accumulate the USDX in the insurance fund
    - Use the accumulated USDX to cover under-collateralized debt in crisis time
    - If the accumulated USDX is not enough, execute originally planed debt auction
- Reasons
    - In result, the balance sheet is same for original/suggested kava minting scenario
    - But the suggested one brings relatively low volatility to kava market price especially in crisis time for crypto assets
- Examples in cryptocurrency market
    - Binance accumulates 10% of their everyday commission earnings to be used for protection of their customers' assets
    - Bitmex Insurance Fund
- Methodologies
    - Accumulated USDX targets certain percentage of entire collateralized debt
    - When acc USDX is far lower than target : kava minted and auctioned faster
    - When acc USDX is near target : kava minted and auctioned slower
    - When acc USDX is over target : acc USDX being auctioned for kava and received kava is burnt ➔ upside market pressure for kava
    - Use long term(1 month or longer) moving average to decide the target value
    - USDX is never minted or burnt in these processes so USDX-debt mapping is not affected by this solution
- Suggested Formulas
    - Target insurance amount : Stress-test methodology
        - assumed price decline : 50%

        - Stress-test : Expected total insolvency amount assuming all collateral price decline 50% and auctioned at -50%, and also assuming all debt auction from insolvency sold by market value
        - Stress-test = Σ MIN[0, insolvency amount of each debt assuming 50% collateral price decline]
        - It is conservative to store entire Stress-test amount in USDX
    - Daily Kava minting amount for insurance fund accumulation
        - Daily Kava minting AMT = (target AMT - current AMT)/100
        - when it is negative, the network burns Kava
        - insurance fund accumulation is dynamically converging to target AMT
        - it takes 69(230) days until the difference narrowed to 50%(10%)

Any applicable appendix

- Binance SAFU : https://www.binance.vision/glossary/secure-asset-fund-for-users
- Bitmex Insurance Fund : https://www.bitmex.com/app/insuranceFund

5.3. Kava CDP Insolvency - CDP/Auction Parameters Review

Background

- CDP global parameters
    - The CDP module has 5 global parameters : CollateralParams, DebtParams, GlobalDebtLimit, SavingsDistributionFrequency, CircuitBreaker
    - CollateralParams has 7 parameters for each collateral asset
    - DebtParams has 5 parameters for each debt asset
- We review each CDP parameter and sub-parameter and analyze possible risk factors and suggest adjustments

Scenario Analysis

- CDP/CollateralParams/LiquidationRatio
    - In traditional financial market, collateralized debt or derivatives products generally have two kinds of collateral value ratio to be used : collateral ratio upon time of 1) debt initiation, and 2) forced collateral liquidation
    - This is because the system wants to provide a systemic buffer between debt raising and liquidation so that the probability of liquidation of collateralized debt in a very short period of time is reasonably low enough.
    - But, Kava Protocol uses same LiquidationRatio for debt initiation, collateral withdraw and forced collateral liquidation

- Auction/BidDuration
    - Price volatility of collateral assets during BidDuration is the reason of bidding price discount for auction participants
    - Longer BidDuration results in heavier bid price discount
    - There is no plan-B for bidder to escape or hedge from such risk because the bidder cannot expect if he/she will win the auction or not

Defining risks

- CDP/CollateralParams/LiquidationRatio
    - No collateral ratio buffer between debt initiation and forced collateral liquidation
    - Usability and comfortness of Kava Protocol users : users will feel that they are not well protected by the system from unreasonably short term liquidation risk
- Auction/BidDuration
    - Long BidDuration period causes heavier bid price discount, resulting in
        - more loss to debt raisers
        - higher risk of CDP insolvency

Suggested Adjustment

- CDP/CollateralParams/LiquidationRatio
    - We suggest to add additional sub-parameter called “InitialRatio” which is used for calculating debt initiation, debt increment and collateral withdraw, but not for forced collateral liquidation
    - It is generally suggested to setup higher ratio for “InitialRatio” than LiquidationRatio so that the Kava Protocol guarantees reasonable buffer between debt/collateral creation/edition and liquidation
- Auction/BidDuration
    - The BidDuration should be short enough to minimize price volatility during the BidDuration period ➔ suggested period : 30 minutes
    - Instead, the auction is extended if the winning bid price is too far away(more than 10%, for example) from current collateral value
    - When auction is extended, bidders should have options to cancel their bids because extension of auction period is not an predictable variable, and such longer period of auction duration is too risky for bidders to bid with market-efficient price
    - The decision of auction rule on highly volatile assets should consider its price volatility and related risk upon auction participants

Any applicable appendix

- CDP/CollateralParams/LiquidationRatio
    - Interactive Brokers(One of the most popular online derivatives broker) margin rules consist of “initial margin” and “maintenance margin”
        - https://www.interactivebrokers.com.hk/en/index.php?f=44472&hm=us&ex=us&rgt=0&rsk=1&pm=0&rst=10100410080801
        - Initial margin is mostly 20~30% higher than maintenance margin
        - Suggested InitialRatio calculation
            - Note that
                - Margin = 1 - (1 / LiquidationRatio)
                - LiquidationRatio = 1 / (1 - Margin)
                - Margin : the collateral/debt ratio becomes 1 when collateral price drops Margin percentage
                - Margin is more intuitive than LiquidationRatio from the perspective of risk measurement
            - Margin for LiquidationRatio(1.5) = (1.5 - 1) / 1.5 = 33.33%
            - Suggested Margin for InitialRatio = 33.33% * 1.3 = 43.33%
            - Suggested InitialRatio = 1 / (1 - 0.4333) = 1.7646

5.4. USDX Price Stability Review

Background

- There exists pegs with different strengths
    - Terra case : strong peg
        - Terra has a functionality called "market swap"
        - It allows any ecosystem participant to swap from token A to token B based on current oracle price
        - It strongly protects the pegging of any stable token in Terra because any deviation of stable token from pegged price will create an arbitrage opportunity via market swap functionality
    - MakerDAO case : weak peg
        - Even though MakerDAO has indirect incentives to buy underpriced DAI or sell overpriced DAI, there is no direct functionality like "market swap" so that any participant without any position can create an arbitrage trade from deviated DAI price
        - Weak peg results in allowing maximum 4% of DAI price deviation

Defining risks

- We expect USDX will have larger deviation than DAI due to lack of strong liquidity and small ecosystem compared to MakerDAO
- Instability of USDX peg represents inability of the stable token to be used as a currency for everyday life

Suggested Adjustment

- We suggest Kava to have daily limited Kava-USDX market swap functionality based on Kava-USD oracle price
- The swap functionality will defend USDX from price deviation in most normal volatility with limited capacity because the deviation of USDX price from USD will create an arbitrage opportunity via market swap functionality
- Daily limit of market swap can be correlated with below environment numbers
    - The appetite of magnitude of stableness of USDX(positive correlation)
    - Accumulated swapped USDX(negative correlation)
    - Entire USDX usage turnover(positive correlation)

Any applicable appendix

- Terra Market Swap : `https://github.com/terra-project/core/tree/develop/x/market`

5.5. Review of “Black Thursday” in MakerDAO

Defining risks

- MakerDAO Black Thursday(12 March, 2020)

Background

- Problems which caused the "Black Thursday"
    - Network congestion ➔ high gas price ➔ official bidding script did not update gas price ➔ no competing bid transactions in auction market ➔ zero price auction result
    - Too small maximum auction lot size or too big minimum auction lot size
    - Too short auction round period
    - Too small number of participants in auction market

Suggested Adjustment

- Kava Network has relatively less risk of network congestion compared to MakerDAO, thanks to Tendermint
- But still, several points are worth being discussed to make the protocol more stable and resilient from market crashes and network congestions
    - The auction-bidding script condition
        - The program should have flexible gas price adjustment feature so that the program allows users to automatically adopt to high gas price situations
        - The program should have easy-to-use UX and configuration so that many participants without technical skills should be able to run the service
    - The auction protocol should allow participations with small capital
        - The auction protocol should have a feature to split the auction into several pieces if the auction amount is too big
        - If the protocol does not have such auction, the auction with larger amount is expected to result in discounted auction price due to too large capital condition
    - The bidDuration should dynamically extended when expected auction price is too far away from market value
    - The auction protocol should allow bidder to bid not based on absolute price, but based on relative discount ratio from oracle price
        - It will minimize the price volatility risk of auction bidders because the bidding price is relative with the oracle price
            - We can expect that most crowded liquidation auction will happen when crypto assets are in huge decline
            - In such crisis, the market price volatility is too high for bidders to take the risk
            - Therefore, it is expected that the auction price will be very discounted
            - Relative discount ratio bidding can minimize such unnecessary risk of bidders
        - It should have a maximum oracle price
            - Which will be used to calculate the necessary collateral to bid the auction
            - If the oracle price on auction closure is above the maximum, the bid is neglected

Any applicable appendix

- MakerDAO Black Thursday discussions
    - https://forum.makerdao.com/t/black-thursday-response-thread/1433

6. Conclusion

Overall in technical review, the implementation is clean, according to Cosmos-SDK, Tendermint patterns as well as good practices about module separation and keeper, kv indexing and iteration

Although some parts of the specification document have not been updated, it has been verified that the main state transition of CDP and Auction is implemented and operated according to the spec. Also test code for each function and scenario is well written, so the code and behavior reliability are high.

In the codebase, B-Harvest found only some Low Severity errors, and no errors or vulnerabilities were found with Medium, High, and Critical Severity. Some suggestions are,

- In order to ensure safety due to the blockchain characteristics, active panic processing is required for unexpected errs
- atomic Commit/PR between specification and code is needed for code update along with lint and code convention for consistency and quality of code

From financial/scenario review, we discovered various issues which are worth deep discussion as below.

- using recent 30 minute average price for debt liquidation
- avoiding debt concentration by time period limits and price range limit
- suggesting Kava Insurance Fund and an algorithm in protocol to manage adequate fund level
- differentiation of LiquidationRatio and InitialRatio for safer environment for users
- reducing price volatility risk for auction participants to make the protocol as efficient as possible
- swap functionality suggested to secure better price stability of stable token, USDX
- review of Black Thursday impact on MakerDAO and lessons from it

We summarized background information from each topic, defined risk factors and their possible outcomes, and suggested adjustments for risk management strategies.

We hope this report finds technical and structural weaknesses of Kava Protocol functionalities and results in constructive evolution of the software and the entire Kava community to possess wiser risk management preparation.

7. References

1. https://github.com/Kava-Labs/kava/tree/8f3858509a0aff6ed26767d35c6ea5f64d808e03
2. Add suggestions from auction code review #486
3. https://github.com/cosmos/cosmos-sdk
4. https://github.com/tendermint/tendermint
5. OWASP Risk Rating Methodology
6. https://www.cmegroup.com/confluence/display/EPICSANDBOX/Nasdaq-100
7. Secure Asset Fund for Users (SAFU) - Definition
8. https://www.bitmex.com/app/insuranceFund
9. https://github.com/terra-project/core/tree/develop/x/market
10. Black Thursday Response Thread - Risk - MakerDAO