Scenario 2 - Redemption TCR drop attack

There is a potential attack of this form:

1. Attacker waits for the system to reach a state with underwater Troves (with CRs < MCR) present, and the underwater debt greater than the LUSD in the SP
2. Attacker redeems an amount of LUSD such that the TCR drops below the CCR and Recovery Mode kicks in
3. Attacker liquidates the underwater Troves, causing some debt remainder to be redistributed
4. The redistributed debt pulls some active Troves below the CCR
5. The attacker deposits LUSD to the SP and liquidates Troves with CR < CCR, earning “extra” collateral surplus and liquidation compensation

The extra liquidations are made possible by the redemption which pulled the TCR below the CCR.

Attacker costs
-Execution costs (gas / Flashbots costs)
-Redemption fee (if attack succeeds)

Attacker revenue
-Attacker’s share of collateral surplus of the liquidated Troves (the initial underwater plus the dragged down), plus all liquidation compensation (0.5% collateral + 200 LUSD per Trove)

Risk for the attacker
The attack can be done in one transaction and via flash loan - no upfront capital is needed. Also, if the attack would result in a loss rather than a profit, the flash loan simply reverts. In this case the only cost to the attacker is the execution cost (gas or Flashbots fee).

Constraints
There are several conditions which must be met for this attack to be feasible:

- There must be underwater Troves in the system
- The system must be in a state such that it is possible to drop the TCR below the CCR via redemption
- Therefore, the redemption itself must have an aggregate CR > CCR
- The underwater Troves must have total debt larger than the total LUSD deposits in the SP
- The net redistributed debt must drag some active Troves down to CR < CCR
- Profit condition: ignoring gas costs, the attacker’s total gains from all liquidations (initial underwater troves, and dragged-down Troves) must be greater than the redemption fee he pays

Can there exist Troves at CR < CCR after redemption?

Why does the attacker need to redistribute debt and coll to create vulnerable Troves - couldn't there already be some Troves at CR < CCR that would become vulnerable after redemption?

It turns out, no - the redemption will have necessarily closed all Troves with CR < CCR, because the redemption itself must have an aggregate CR > CCR (see the proof below - section 1). This means that some of the Troves in the redemption must sit above the CCR, and some below, which in turn means that at least all Troves with MCR < CR < CCR must be redeemed. Therefore, the only remaining Troves after the redemption have CR > CCR.

1. Formal conditions for TCR drop below CCR

Let's separate the system's Troves into 3 categories:

A: Underwater Troves at CR < MCR (coll: c_A, debt: d_A)
B: Troves hit in the redemption (coll: c_B, debt: c_B)
C: Remaining active Troves at CR > MCR after redemption (coll: c_C, debt: d_C)

We then have these pre- and post-conditions on the TCR:

Pre-condition - TCR above CCR

1) $(c_A + c_B + c_C) / (d_A + d_B + d_C) > CCR$

Post-condition - TCR below CCR

2) $(c_A + c_C) / (d_A + d_C) < CCR$

That is, after the redemption, all collateral and debt in group B is removed.

Underwater condition - aggregate CR of group A < MCR

3) $c_A/d_A < MCR$

Condition on CR_B

We can derive the condition on the aggregate CR of the redeemed Troves, CR_B. Rewriting 1) gives:

4) $c_A + c_C > CCR\,(d_A + d_B + d_C) - c_B$

And rewriting 2) gives:

5) $c_A + c_C < CCR(d_A + d_C)$

And using 4) in 5) gives:

6) $CCR\ (d_A + d_B + d_C)\ - c_B < CCR(d_A + d_C)$

Which simplifies to:

7) $CCR\ d_B\ - c_B\ < 0$

That is:

9) $c_B/d_B > CCR$

Yielding the condition on CR_B.

2. Condition on the redeemed amount

Can we say anything about how much must be redeemed (d_B), given some system state, in order to drop the TCR below the CCR?

Is there some relationship between d_B and d_A, d_C, and the TCR?. If there are no underwater Troves, redemptions always improve the TCR, so we know d_A must be non-zero.

It seems likely that all else equal, the higher the TCR, the greater d_B must be to drop it below CCR.

A clear relationship seems tricky to establish though. From 9), we know that the redemption must be greater than the CCR to drop the TCR below CCR. However, this can only be the case if the *distribution* of collateral and debt across Troves is favorable.

We can try to formalize this - let's rewrite inequalities 1) and 2) as equations, and derive an expression for d_B. We assume the TCR starts at some buffer **e** above the CCR before redemption, and the redemption drops it down to the CCR.

So 1) becomes:

10) $(c_A + c_B + c_C) / (d_A + d_B + d_C)\ = CCR + e$

And 2) becomes:

11) $(c_A + c_C) / (d_A + d_C)\ = CCR$

i.e.

12) $c_A + c_C = CCR\ (d_A + d_C)$

Rearranging 10):

13) $c_B - d_B(CCR + e) = (CCR + e)(d_A + d_C) - (c_A + c_C)$

Substituting 12):

14) $c_B - d_B(CCR + e) = (CCR + e)(d_A + d_C) - CCR\ (d_A + d_C)$

I.e.

15) $c_B - d_B(CCR + e) = e(d_A + d_C)$

Now, rewrite c_B using the collateral ratio of B, CR_B:

16) $CR_Bd_B - d_B(CCR + e) = e(d_A + d_C)$

We can also rewrite $(d_A + d_C)$ as the total system debt before redemption, D, less the redeemed amount d_B:

17) $CR_Bd_B - d_B(CCR + e) = e(D - d_B)$

So:

18) $d_B(CR_B - CCR) = eD$

Therefore we have an expression for d_B:

19) $d_B = eD/(CR_B - CCR)$

What does this tell us? All else equal:

- The greater the total debt D the larger the redemption must be
- The greater the TCR buffer above CCR (e), the larger the redemption must be
- The greater the underwater debt d_A, the larger the redemption must be (since $D = d_A + d_B + d_C$)

However, there is also an inverse relationship between the redeemed amount d_B and the collateral ratio of the redeemed amount CR_B. The greater the CR_B, the less debt needs to be redeemed to drop the TCR.

Hence the redeemed amount needed to drop the system into RM depends on the **distribution** of debt and collateral: for a given redeemable amount, the higher the CR_B of it, the more the TCR will drop when it is removed.

This dependence on the distribution makes it hard to analyse. However for a given TCR, system size D and underwater debt d_A, we can find an upper bound on CR_B and thus on d_B (see Section 5).

3. Which Troves get dragged down below the CCR at redistribution?

Redistributions don’t always cause drag downs. The remaining active Troves can only be dragged down below the CCR by redistribution if certain conditions are met. And only some of those Troves could possibly be dragged down.

We can try to derive a “drag down” condition for an active Trove relating its collateral to its CR, the CCR and the aggregate CR of the redistributed Troves, the distributed collateral, and the total collateral C.

Lets define terms:

c: collateral of active Trove
d: debt of active Trove
cr_A: collateral ratio of active Trove

x: collateral of liquidated Troves
y: debt of liquidated Troves
C: collateral of entire system
C’: collateral of entire system less collateral of redistributed Troves (i.e. C - x)

Redistributions distribute collateral and debt of the liquidated Troves proportional to the recipient Trove’s share of C’. Therefore we have the drag down condition:

20) $(c + xc/C') / (d + yc/C') < CCR$

So:

21) $(c + xc/C') < CCR\ (d + yc/C')$

And rewriting the active Trove’s debt d in terms of its collateral c and collateral ratio cr_A:

22) $(c + xc/C') < CCR\ (cr_A/c + yc/C')$

I.e.

22) $c + xc/C' - CCR\ yc/C' < CCR\ cr_A/c$

23)$c\text{^}2(1 + (x - CCR\ y)/C') < CCR\ cr_A$

24) $c\text{^}2 < CCR\ cr_A / (1 + (x - CCRy)/C')$

We rewrite C’ in terms of total system collateral C and liquidated collateral x, and take the square root, discarding the negative value.

25) $c < \text{sqrt}(CCR\ cr_A / (1 + (x - CCRy)/(C\text{-}x))$

Now let’s flip the inequality to give us the condition for a Trove *not* being dragged down:

26) $c >= \text{sqrt}(CCR\ cr_A / (1 + (x - CCRy)/(C\text{-}x))$

Therefore for some given system size C and liquidated coll x and debt y, we have a non-linear relationship between the active Trove’s collateral and its collateral ratio.

The larger a Trove’s collateral ratio, the smaller c can be while keeping the Trove safe from drag down.

However, this condition isn’t super informative for users - it still has (non-linear) dependencies on size of the redistributed Troves and the system size.

The redistributed amount in particular is hard to estimate, though if we could upper-bound it we could come up with a safety metric for Trove owners.

4. Profit conditions

We can derive a formula for the profit condition, making some assumptions along the way.

For a given redeemed amount d_B, the attacker’s pays the redemption fee as costs:

27) attack_cost = redemption_fee

Their revenue is the sum of SP gains and gas compensation from the first liquidation (which partially hits the SP and remainder is distributed), and then SP gains and gas costs from the second liquidation of the dragged down Troves.

We assume gas costs and the 200 LUSD gas compensation are small enough to be ignored.

Let’s define terms:

D: total system debt
D’: total system debt after liquidation 1 and redemption
S: total LUSD in SP before liquidation 1
d_A: total underwater debt before attack i,e. with $CR_A <= MCR$
d_B: total redeemed amount
k: Initial ratio of SP to total debt
s: Attacker’s share of SP before liquidation 1
f: Initial ratio of underwater debt to total debt, i.e. $0<f<=1$
p: Fraction of total debt redeemed by attacker
h: Fraction of outstanding LUSD supply that attacker can buy/loan and deposit to SP after liquidation 1 and redemption

q: Fraction of remaining collateral that can be dragged down by redistribution

Their gross gain is given by:

28) gain = liq_1_SP_gain + liq_1_gas_comp + liq_2_SP_gain + liq_2_gas_compensation

Also, the attack is subject to the constraint that redistribution occurs, i.e. d_A > S, that is:

29) fD > kD

i.e.

30) f > k.

The liq_1_SP_gain is the collateral surplus from liquidation 1.

<u>Assumption</u>: liquidated Troves have CR = MCR, and therefore there’s a collateral surplus of 0.1 * debt.

And since the entire SP debt (S) must be liquidated, the attacker earns:

31) liq_1_SP_gain = 0.1sS

His gas compensation (0.5% of the entire collateral liquidated) is:

32) liq_1_gas_comp = 1.1fD/200

The remaining LUSD supply is the initial total debt less the amount liquidated against the SP and the redemption.

That is:

33) D’ = D - S - d_B

i.e:

34) D’ = D(1- k - p)

This is the basis for both liq_2_SP_gain and liq_2_gas_comp. Now, we parameterize it and say some fraction q of this debt can be dragged down by redistribution.

Also, the attacker holds some fraction h of this and redeposits it to the SP.

Now, the amount liquidated against the SP is the lesser of the SP size and the liquidateable Trove size. And the attacker’s gain is only the collateral surplus on this. That is:

35) liq_2_SP_gain = 0.1(min(qD(1-k-p), hD(1-k-p))

*(Issue: there could be *another* redistribution here, if liquidated debt is greater than the SP size - however any chain reaction of redistribution and further drag-downs should be very self limiting)*

The attacker's gas compensation is based on the total collateral liquidated:

36) $liq\_2\_gas\_comp = 1.1qD(1- k - p)/200$

Putting it all together:

37) $attack\_gain = liq\_1\_SP\_gain + liq\_1\_gas\_comp + liq\_2\_SP\_gain + liq\_2\_gas\_compensation$

38) $attack\_gain = 0.1sS + 1.1fD/200 + 0.1(min(fD-S, hD(1-k-p)) + 1.1qD(1-k-p)/200$

i.e.

39) $gain = 0.1skD + 1.1fD/200 + 0.1(min(qD(1-k-p), hD(1-k-p)) + 1.1qD(1-k-p)/200$

Extracting D and simplifying:

40) $attack\_gain = D[0.1sk + 1.1f/200 + 0.1min(q, h)(1-k-p)+ 1.1q(1-k-p)/200]$

Returning to the attack cost, and assuming the baseRate starts at 0.5%, the attacker pays the redemption fee, i.e:

41) $attack\_cost = d_B* (1/200 + d_B/2D) = pD/200 + p^2D^2/2D = pD/200 + p^2D/2$

So we have the profit equation:

42) $profit = D [0.1sk + 1.1f/200 + 0.1min(q, h)(1-k-p) + 1.1q(1-k-p)/200 - p/200 - p^2/2]$

We can at least now plug in different values for s, k, f, h and p and see whether the attacker can profit or not: https://docs.google.com/spreadsheets/d/1z0lusYAXuWbnfGzzNuDsyar5Br2CU9Xo7WHk25i4qJ8/edit?usp=sharing

5. Upper bound on / worst case of CR_B

It's clear from 19) that the "best case" distribution is when CR_B is low: in the limit as $CR_B \rightarrow CCR$, the redemption size d_B needed to drop the TCR explodes. Since d_B is a monotonic function of CR_B, the "worst case" d_B occurs when CR_B is at its maximum.

What is the maximum value of CR_B? Let's consider the collateral ratio of of the active Troves, i.e. the total system less the collateral and debt of the underwater Troves:

43) $CR_{Active} = (C - d_A) / (D - d_A)$

By definition, CR_B cannot be greater than this value, since redemptions start from the active Troves with lowest collateral ratio. The greatest value CR_B can take is CR_{Active}. If all active Troves have the same collateral ratio, then the CR of the redeemed amount is always CR_{Active}, regardless of the redeemed amount.

44) $CR_B <= (C - c_A) / (D - d_A)$

And worst case:

45) $CR_B = (C - d_A) / (D - d_A)$.

Now, lets plug this into 19):

46) $d_B = eD/((C - c_A) / (D - d_A) - CCR)$

And rearranging:

47) $d_B = eD(D - d_A) /(C - c_A - CCR(D - d_A))$

Rewriting the underwater collateral in terms of the underwater debt:

48) $d_B = eD(D - d_A) /(C - CR_A d_A - CCR(D - d_A))$

And rewriting total collateral in terms of initial TCR and total debt, and e in terms of initial TCR:

49) $d_B = (TCR - CCR)D(D - d_A) / (D(TCR) - CR_A d_A - CCR(D - d_A))$

So

50) $d_B/D = (TCR - CCR)(D - d_A) / (D(TCR) - CR_A d_A - CCR(D - d_A))$

We now have an expression for **the worst-case percentage of total system debt that must be redeemed to drag the TCR down to the CCR, given some initial TCR, system size and underwater debt.**

Let's plot it:
https://www.desmos.com/calculator/x5gir4xrhu

We see that:

- Larger d_A reduces the redemption d_B required

- Lower CR_A slightly reduces the redemption d_B required

- A greater TCR increases the redemption d_B required

This third point is crucial. We see that even for large underwater debt (e.g. when d_A is 20% of system size), then a TCR of just 155% entails a necessary redemption of 30% of system size. Plugging this into the profitability table, we see that for this redeemed amount an attack would be unprofitable even if the attacker controlled the entire Stability Pool.

This relationship between TCR and d_B looks promising - it seems even at low TCRs, very large redemptions (and thus large fees) are needed.

Also, this is the worst case collateral distribution which is very unrealistic. In practice, CR_B would be lower, And the needed redemption size dB would be higher.

The crucial metric is dA/D, i.e. the fraction of system debt that is underwater. If this is very large, e.g. 50%, then for low TCRs, e.g. TCR<200%, attacks can be profitable.

However, such huge underwater debt seems unlikely, especially at high TCRs >200%.

In fact, we can combine the minimum redemption size 50) with the profit equation 42), and plot a single graph for profitability as a function of system state params: TCR, CCR, CR_A, s, k, f, h and p:
https://www.desmos.com/calculator/nagh1cwanj

Attack variant: Attacker creates the underwater debt

To create the underwater debt themselves, the attacker must first create a Trove at CR == MCR and wait for the ETH price to drop. This attack would likely be done via Flashbots sandwich, since they must now frontrun the ETH price drop with their openTrove transaction. They consequently bear the risk associated with that - e.g. re-org risk. This exposes them to the slight chance of an outright net loss equal to 10% of the Trove's collateral, e.g. if someone else liquidates them before they do the redemption.

(We assume they open a Trove at CR == MCR and an infinitesimal ETH price drop occurs, dragging it just below MCR)

If they succeed, their profit margin would be lower than otherwise since even if they own the whole initial SP, part of their liquidated Trove must get redistributed.

Attacker gains

The attacker now no longer profits from liquidation 1 since the underwater debt is all theirs. In fact they lose the collateral surplus, but regain part of it through their initial position in the SP and the gas compensation.

So their profit/loss from liquidation 1 is:

-0.1fD + 0.1skD + 1.1fD/200

They also pay the borrow fee on the Trove they open , i.e. fD/200 (Hence, the gas compensation on the underwater debt cancels the borrow fee).

The adapted profit equation is then:

53) profit_worst_case_attacker_creates_underwater_debt = D [0.1sk -0.1f + 1.1f/200 + 0.1min(q, h)(1-k-p) + 1.1q(1-k-p)/200 - p/200 - p^2/2 - f/200]

i.e.

54) profit_worst_case_attacker_creates_underwater_debt = D [0.1sk - 0.1f + 0.1f/200 + 0.1min(q, h)(1-k-p) + 1.1q(1-k-p)/200 - p/200 - p^2/2]

Plotting it:
https://www.desmos.com/calculator/nttg0kwdos

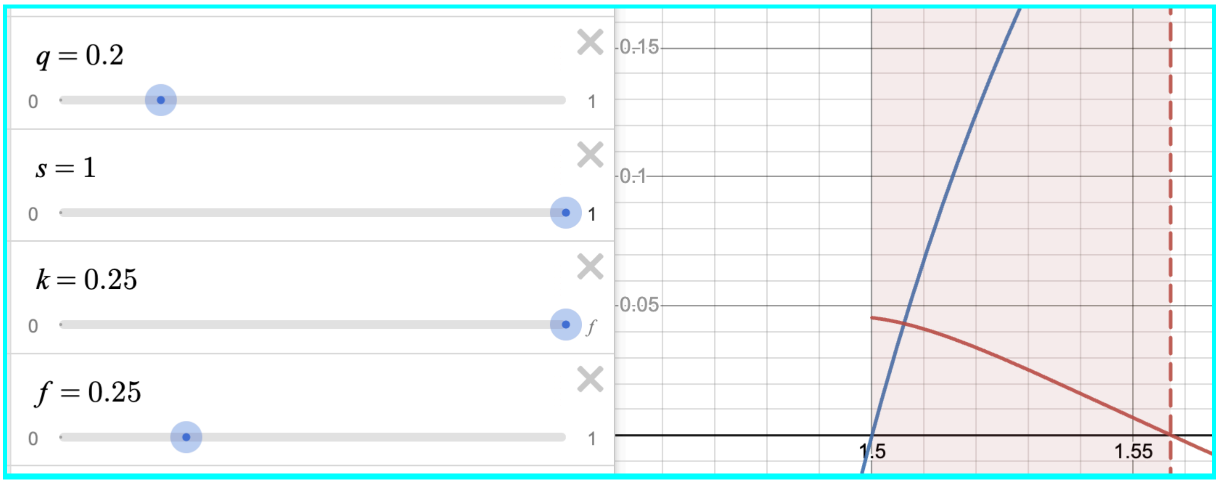
Analysis

Let's choose a value of q=0.1 i.e. 10% of remaining debt gets dragged down below CCR, which seems conservative (i.e. unlikely and high).

Comparing this attacker-created underwater debt variant to the original, we see that:

-The profitable range is smaller
-Profits are lower
-The attacker must obtain a high initial share of the SP (s) in order to profit at all

Attacker uses existing underwater debt:



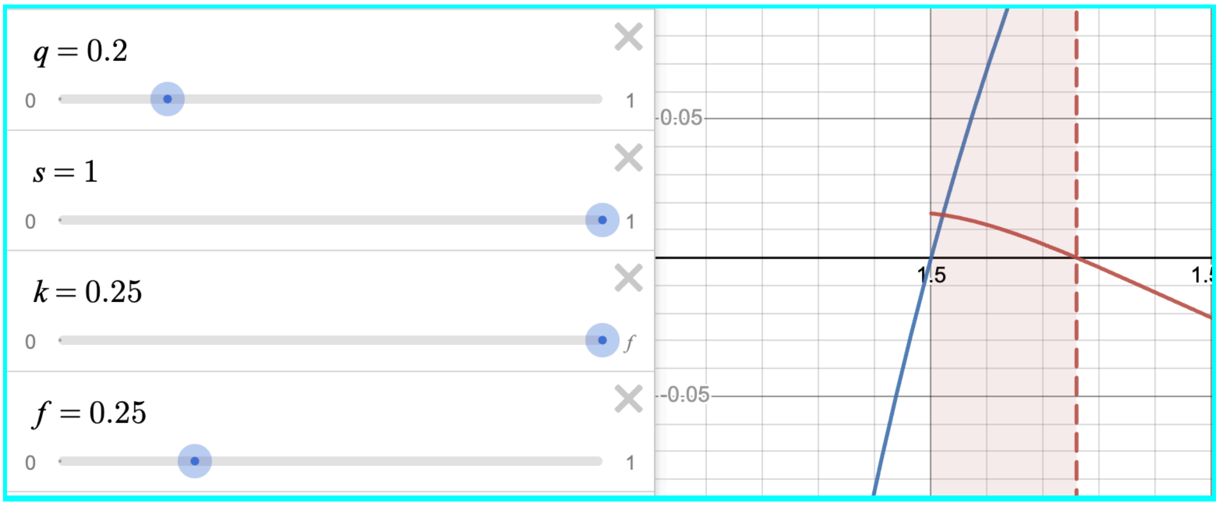
$q = 0.2$

$s = 1$

$k = 0.25$

$f = 0.25$

Attacker creates underwater debt:



$q = 0.2$

$s = 1$

$k = 0.25$

$f = 0.25$

To make the attack profitable at higher TCRs (i.e. TCR >160%), the attacker must create large underwater debt (>74% of total debt), with a similar fraction of total debt in the SP, and the attacker owning most of the SP (i.e. >95%). The attacker's profit is relatively low here, e.g. 0.3% of total debt.

Still, it is a profitable attack, and it is concerning that the attacker can in principle create any amount of underwater debt they like, rather than relying on external conditions.

If the potential dragged down fraction q is lower though, the attack is only (slightly) profitable up to a TCR of ~159%.

TODO: determine how much of the remaining debt after redemption can realistically be dragged down by a large redistribution.

7. Conclusions

With existing underwater debt

As long as the TCR is not very low (e.g. ~150%) or the underwater debt is not a large chunk of the total debt (e.g. d_B > 30%), attacks seem unprofitable. Also, if the TCR is very low and close to 150%, then it's already at high risk of entering Recovery Mode due to an ETH price drop.

These conclusion are true under the very unrealistic worst-case assumptions for the collateral ratios and distributions of the redeemed amount. It also assumes an empty initial SP (which maximises profit) and that the attacker owns the entire remaining LUSD supply after the redistribution.

In practice, these assumptions won't hold and all else equal the profitable threshold will be even closer to the CCR.

We also clearly see that as long as the LUSD in the SP fully covers the debt of the underwater Troves, a profitable attack is not possible regardless of the redeemed amount.

Attacker creates underwater debt

If the attacker creates the underwater debt themselves, then profit is possible at higher TCRs, however the profit heavily depends on q, the fraction of remaining debt that can be dragged down below 150% by redistribution.

This fraction depends on the particular distribution of Troves. And the redeemed amount still assumes a worst-case distribution - so in practice, profits/profitable TCR range will be even lower.

Here the attacker must also sandwich an ETH price drop, and likely run a re-org risk from using a Flashbots bundle.

Summary

Both attack variants rely on very specific and unlikely “stars aligned” unhealthy system states.

Additionally, borrowers who maintain CRs of 150% will remain immune to liquidations even in Recovery Mode (unless they’re unlucky enough to get dragged down below 150% by the redistribution, but that’s a low risk for a realistic drag down fraction q).

Suggested further work

- Derive an upper bound on the total debt that can be dragged down as a function of redistribution size
- Derive an upper bound on the redistributed Trove size as a % of system size