Dutch Liquidation Analysis

Michael A. Bentley

May 14, 2024

1 Introduction

The liquidation bonus on Euler is dependent on health score. As health declines below 1, the liquidation bonus increases. This mechanism acts as a quasi-Dutch auction, helping to find a fair market cost for liquidations. On Euler v1, the liquidation bonus for loans of $> \$10,000$ averaged just $\sim$0.77% (see Messari article). For larger loans, it was even smaller. This made Euler v1 a more attractive protocol to borrow on than other lending protocols, where liquidation bonuses, even for large loans, tend to be much larger and more punitive for borrowers.

As Euler approaches relaunch with its new Euler Vault Kit, its liquidation scheme was recently re-analysed in depth by four security partners: ChainSecurity, OpenZeppelin, Spearbit, and Omniscia. Each raised interesting points about trade-offs and potential attack vectors associated with the mechanism. These are considered in detail below, starting with an overall discussion of the mechanism in the section dedicated to observations from ChainSecurity.

2 ChainSecurity

ChainSecurity considers issues around when it might become more profitable for a liquidator to do multiple partial liquidations than a single large one. This happens in circumstances when the liquidator's dynamic bonus percentage increases (rather than decreases) with each new liquidation. This can be considered as a liquidation optimisation on the part of a liquidator, but it can potentially lead to increased bad debt for the protocol.

The liquidation bonus in Euler is dependent on health score. Broadly speaking, there are two possibilities for what happens when a position is liquidated. In most cases, the health of a position will *increase* after liquidation (the whole point of liquidating a position). A liquidation which improves health will decrease the bonus on future liquidations, meaning it is always in the interests of liquidators to do a single liquidation than many partial ones.

However, as was noted in the OpenZeppelin audit of Compound v2, a small subset of liquidations can actually *decrease* the health of a position (see section on Counterproductive Incentives here). This is not something that can generally be fixed (although see Omniscia idea below). It happens because at some point a borrower has insufficient excess collateral to be able to payout as a bonus. So every further liquidation takes them closer towards insolvency. The interesting side effect of this behaviour on Euler, however, is that liquidation which lowers health will increase the bonus on future liquidations in Euler, meaning it is always in the interests of liquidators to break a liquidation down into as many partial liquidations as possible to achieve the maximum net bonus possible.

2.1 When does health decline?

Under what circumstances can a liquidation decrease the health score of a user? Let the market value of a user's collateral and debt at time t be defined, respectively, as $c(t)$ and $d(t)$. Then a user's health at time t be defined as

$$h(t) = \frac{c(t)}{d(t)} \cdot v, \tag{1}$$

where $v \in [0, 1]$ is the loan-to-value (LTV) parameter specifying how over-collateralised a user needs to be before they are eligible for liquidation.

A user is eligible for liquidation when $h(t) < 1$, which means that a user's risk-adjusted collateral value is lower than their outstanding debt:

$$c(t) \cdot v < d(t). \tag{2}$$

When a user is eligible for liquidation, a third-party liquidator is rewarded for processing a repayment of their debt with a bonus, b. In Euler, this bonus depends on a user's health score. As the user's health score declines, the bonus grows, according to the equation

$$b(t) = 1 - h(t) = 1 - \frac{c(t)}{d(t)} \cdot v. \tag{3}$$

Suppose a user's initial health score at time $t = t_0$ is $h(t_0)$, then after liquidation at time $t = t_1$, in which an amount a of the user's debt is repaid, the user's health score is

$$h(t_1) = \frac{c(t_1)}{d(t_1)} \cdot v = \frac{c(t_0) - (1 + b(t_0))a}{d(t_0) - a} \cdot v = \frac{c(t_0) - \left(2 - \frac{c(t_0)}{d(t_0)} \cdot v\right) a}{d(t_0) - a} \cdot v. \tag{4}$$

When does a user's health score decrease after liquidation? This happens when $h(t_1) < h(t_0)$, which, after simplification, can be found to be when

$$b(t_0) > \frac{c(t_0)}{d(t_0)} - 1. \tag{5}$$

Intuitively, a user's health score is lowered after liquidation if the bonus is greater than the relative over-collateralisation of the position. For example, if a user is 10% over-collateralised, but the bonus is 11%, then a user's health score declines after liquidation. For any bonus less than 10%, a user's health will always increase after liquidation.

Alternatively, equation (5) can be re-written in a way that relates the LTV of the user to the required LTV of a collateral-debt-asset pair, v, as

$$\frac{d(t_0)}{c(t_0)} < \frac{1 + v}{2}. \tag{6}$$

2.2 Liquidation trajectory

We can think of a user's position at time t as a point $p(t) = (d(t), c(t))$ in the debt-collateral plane (dc-plane). What trajectory does a user's point follow as they are liquidated? In general, it will be a path that takes their position down and to the left of the plane, roughly in the direction of the origin. The gradient of the path will depend on the bonus received by the liquidator for a particular liquidation. For a liquidation amount a, the gradient is given by

$$\frac{\Delta c}{\Delta d} = \frac{c(t_1) - c(t_0)}{d(t_1) - d(t_0)} = \frac{c(t_0) - (1 + b(t_0))a - c(t_0)}{d(t_0) - a - d(t_0)} = 1 + b(t_0). \tag{7}$$

If the liquidator does a single large liquidation for the entire debt amount, the liquidation trajectory is just a straight line with slope $1 + b(t_0)$.

If, however, the liquidator chooses a smaller repayment amount that enables them to perform many partial liquidations, the liquidation trajectory will not be a straight line, because the liquidation bonus, $b(t)$, grows (under the circumstances outlined above) with every incremental liquidation.

In the absence of any other constraints, the best-case scenario for the liquidator in terms of maximising their overall bonus will be to do infinitely many small partial liquidations, letting the bonus grow a infinitesimal amount with each one. In this case, we can find the trajectory of liquidation using methods from differential calculus. We have the initial gradient of the liquidation trajectory at the point $p(t_0) = (c(t_0), d(t_0))$ as

$$c'(c, d) = 1 + b(t_0) = 2 - \frac{c}{d} \cdot v, \tag{8}$$

where we have simply substituted in for the liquidation bonus $b(t_0)$ from equation (3). This is a first-order linear ordinary differential equation (ODE). It can be solved for the liquidation trajectory c by using an integrating factor (or just asking Wolfram Alpha). We have solution for the ODE of:

$$c = C_1 d^{-v} + \frac{2d}{1 + v}, \tag{9}$$

where C_1 is factor of integration. We can solve for C_1 using the initial condition $p(t_0) = (c(t_0), d(t_0))$, giving $C_1 = \left(c_0 - \frac{2d_0}{1+v}\right) d_0^v$.

This solution assumes that the liquidation bonus can grow arbitrarily large during liquidation. In practice, however, Euler uses a maximum bonus $b_{\max}$, which constrains the liquidator. Thus the gradient of equation (9) can never exceed $1 + b_{\max}$ and the true solution must be piece-wise. For the maximum bonus, we have a straight line defined by an equation

$$c = C_2 + (1 + b_{\max})d, \tag{10}$$

where C_2 is an arbitrary constant that must be chosen to satisfy the goal of joining a straight line liquidation trajectory (defined by the maximum bonus) to the curved trajectory defined by equation (9). To choose this constant, we first need to identify where the curved trajectory has gradient $1 + b_{\max}$. Differentiating equation (9) with respect to d, we have

$$c' = -vC_1 d^{-(1+v)} + \frac{2}{1+v}. \tag{11}$$

We then set $c' = 1 + b_{\max}$ and solve for the coordinate $d = d_{\max}$ at which the bonus equals the maximum available. We obtain

$$d_{\max} = \left(\frac{vC_1}{\frac{2}{1+v} - (1 + b_{\max})} \right)^{(1+v)^{-1}}. \tag{12}$$

We then substitute back into equation (9) for $d = d_{\max}$ to find the coordinate $c = c_{\max}$ at which the bonus equals the maximum available. For this, we have

$$c_{\max} = C_1 d_{\max}^{-v} + \frac{2d_{\max}}{1+v}. \tag{13}$$

Finally, we can substitute into equation (14) for $d = d_{\max}$ and $c = c_{\max}$ and solve for the arbitrary constant C_2, to obtain

$$C_2 = c_{\max} - (1 + b_{\max}) d_{\max}. \tag{14}$$

The final piece-wise solution of the liquidation trajectory is therefore given by

$$c = \begin{cases} C_2 + (1 + b_{max})\, d & \text{for } 0 \leq x < d_{max} \\ C_1 d^{-v} + \dfrac{2d}{1+v} & \text{for } d_{max} \leq x \leq d_0. \end{cases} \tag{15}$$

2.3 Visualisation

The equations derived here can all be visualised on the dc-plane as in Fig. 1, which shows an example in which a position is liquidated for a higher overall bonus when a liquidator uses many partial liquidations (purple trajectory) compared to when they use a single larger one (blue trajectory).

2.3.1 How to interpret the figure

Green area: delineates space on top where user is fully solvent and their health score is above 1, meaning they are ineligible for liquidation.

Red area: delineates space below where the user is already insolvent, meaning they have more debt than collateral and remain eligible for liquidation until all their collateral has been removed.

Orange area: delineates space in the middle, where the user has a health score below 1, but they still have more collateral than debt. They are solvent, but eligible for liquidation.

Orange line: delineates where health can decline after liquidation. A user position above the orange line will always have their health score improved by liquidation, whereas a user position below the orange line may have their health reduced by liquidation.

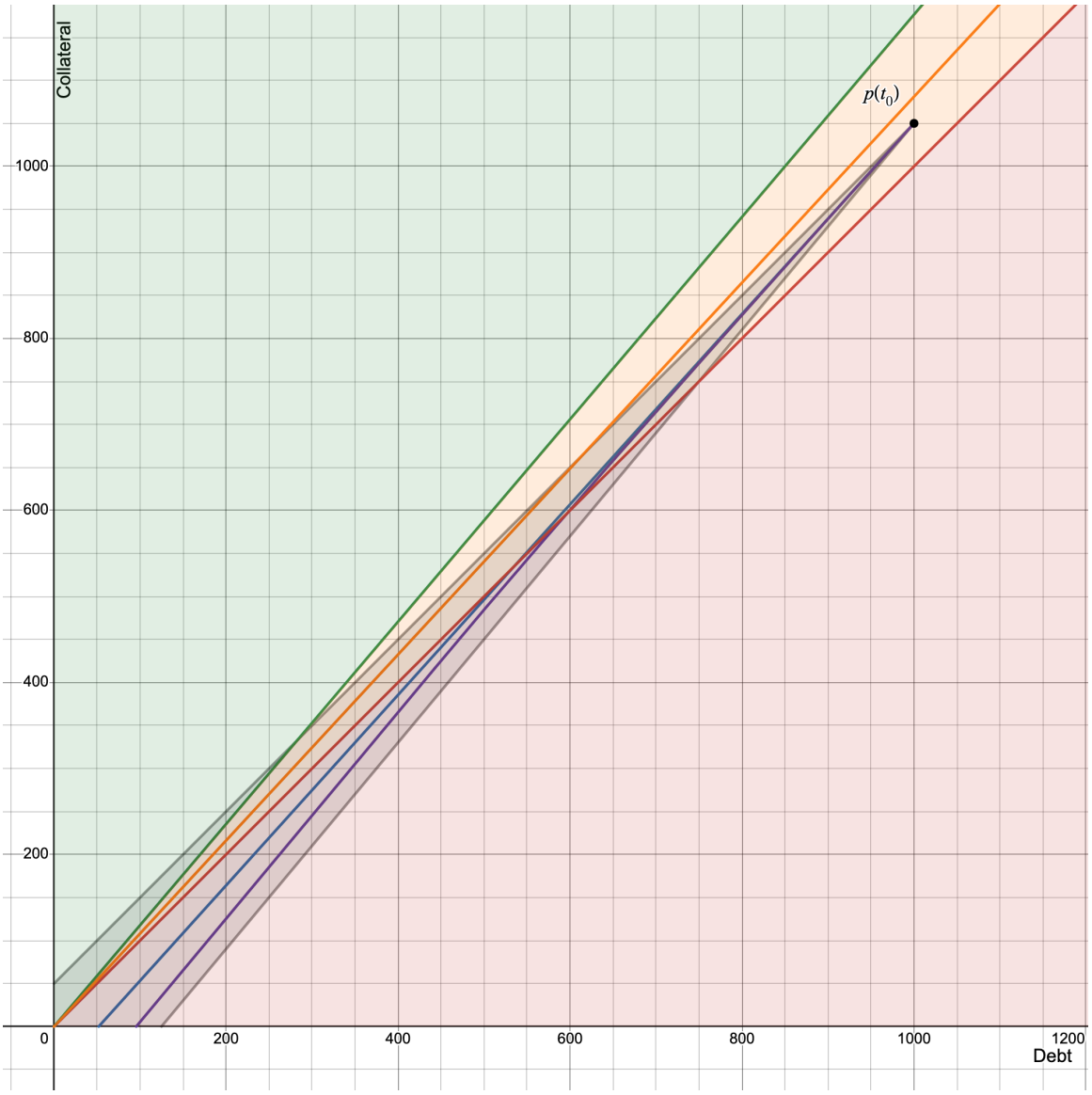

Figure 1: **Multiple partial liquidations can be represented by a trajectory on the debt-collateral plane**. For the given parameters, an example of position being liquidated using many smaller partial liquidations (purple trajectory) compared to single larger liquidation (blue trajectory) is shown. Parameters: $v = 0.85, b_{\max} = 0.2, c_0 = 1050, d_0 = 1000$. Variables: $h(t_0) = 0.8925, b(t_0) = 0.1075$. Desmos graph here.

Grey lines: delineate the space of possible liquidation trajectories. The top grey line represents the liquidation trajectory for a liquidation bonus of zero. The lower grey line represents the liquidation trajectory for the maximum liquidation bonus.

Blue line: this is the liquidation trajectory experienced by a user who's entire position is liquidated in its entirety using the liquidation bonus calculated from their health score.

Purple line: this is the liquidation trajectory that represents what happens when a user is subject to infinitely many small partial liquidations, in which the the liquidation bonus grows a infinitesimal amount with each one. This is the best-case scenario for a liquidator and worst-case scenario for the protocol, because it leads to greater generation of bad debt.

2.4 Conclusions

Liquidations that drive users towards bad debt are a mathematically unavoidable outcome for any lending protocol that relies on liquidation bonuses to secure liquidations in a decentralised manner. Euler is not unique in having liquidations which might drive users towards the creation of bad debt, but ChainSecurity is right to highlight how its dynamic liquidation incentives do create the unique possibility for liquidators to profit from breaking liquidations into many small pieces.

The general consensus is that this behavior cannot be used to exploit the system and therefore needs to be documented, but not necessarily remediated. Liquidators can, under certain conditions, win a higher bonus if they do smaller liquidations, but the bonus is always capped at some maximum. Inspection of the Desmos simulation in Fig. 1 shows that there is a relatively small parameter space in which multiple partial liquidations is ever profitable for a liquidator. Given that static liquidation incentives leak large amounts of collateral value out of a system unnecessarily, and dynamic liquidation incentives are empirically proven on Euler v1 to lower the cost of liquidations to lenders and borrowers, it seems highly likely dynamic liquidation incentives remain a net positive benefit for borrowers and those lending to them in most scenarios, even if there is the potential for increased liquidated collateral and bad debt through partial liquidations in some rare cases.

It should also be noted that in those rare cases in which multiple partial liquidations are possible, the liquidation trajectory derived herein is a worst-case scenario for the protocol, and unlikely to be attained in reality. Practical computational considerations apply, which will mean a liquidator cannot perform infinitely many small liquidations. More likely is that they will break a liquidation up into a fixed number of n chunks and perform it in a batch transaction. However, as the number of chunks n grows large, gas costs and gas limits start to constrain a liquidator's ability to profit from chunking.

3 OpenZeppelin & Spearbit

OpenZeppelin and Spearbit consider issues around sandwich attacks for price oracle updates and their interaction with dynamic liquidation incentives, especially in the context where a pull-based oracle is used. They identify some rare circumstances in which a malicious actor may be able to sandwich a price oracle update in between a flashloan + ordinary loan and self-liquidation transaction in order to profit. These sandwich attacks are not necessarily unique to the Euler mechanism, but may be made more likely for pull-based oracles, which the Euler Vault Kit is assumed to support. We consider the mechanics of the flashloan attack on collateral price decreases identified by security researchers, and extend their analysis to consider debt asset price increases, as well as multi-block attacks that do not require use of a flashloan.

3.1 Flashloan sandwich attack

For a flashloan attack the steps are:

- Identify when a price update is coming
- Flash loan collateral and max borrow before price update
- Price update happens (possibly performed by malicious actor themselves in the event of pull-based oracle)
- Self-liquidate the max borrow position, retrieving all collateral and repaying only a portion of the borrow
- Repay the flashloan using retrieved collateral
- Keep the remaining borrowed assets as profit

The attack is profitable when the entire collateral balance is seized (to repay the flashloan) while repaying fewer debt assets than assets that were borrowed for the ordinary loan. Let us explore the mechanics of this idea in more depth.

3.1.1 Mechanics

The attacker begins at time t_0 by flash loaning $c(t_0)$ worth of collateral and then using it to borrow $d(t_0) = c(t_0)v$ worth of some liability asset. Their health at this time is

$$h(t_0) = \frac{c(t_0)}{d(t_0)} \cdot v = \frac{c(t_0)}{c(t_0)v} \cdot v = 1. \tag{16}$$

What they do next depends on whether or not there is a price decrease of collateral or a price increase of debt asset.

3.1.2 Collateral price decrease

If a price update lowers the value of their collateral at time t_1 by a factor of $p_c \in [0, 1]$, then $d(t_1) = p_c c(t_0)v$ and their health is now

$$h(t_1) = \frac{c(t_1)}{d(t_1)} \cdot v = \frac{p_c c(t_0)}{c(t_0)v} \cdot v = p_c. \tag{17}$$

As the user's health score has declined, the bonus percentage available to themselves for a self-liquidation is

$$b(t_1) = 1 - h(t_1) = 1 - p_c. \tag{18}$$

The malicious actor can therefore self-liquidate, receiving the entire collateral amount to their wallet. This total amount comprises a portion

$$c_b(t_1) = b(t_1)c(t_0) = (1 - p_c)c(t_0), \tag{19}$$

which is their liquidation bonus, and a remaining portion for repaying debt of

$$c_r(t_1) = c(t_0) - (1 - p_c)c(t_0) = p_c c(t_0), \tag{20}$$

which is the amount set aside for repaying of the borrowed asset. We must remember, however, that this amount is now worth less than it once was, by a factor of p_c, so the total amount of debt required to be repaid by the self-liquidator for $c_r(t_1)$ units of collateral is

$$d_r(t_1) = p_c c_r(t_1) = p_c^2 c(t_0). \tag{21}$$

After repaying their flash loan using the collateral, and the ordinary loan using the assets they borrowed, there may be remaining debt on the protocol. This is the same value as the attacker's profit, which is given by

$$w(t_1) = d_r(t_1) - r(t_1) = c(t_0)v - p_c^2 c(t_0) = c(t_0)(v - p_c^2). \tag{22}$$

The malicious actor therefore profits from this sandwich attack when $w(t_1) > 0$, which implies

$$\begin{aligned} &c(t_0)(v - p_c^2) > 0 \\ \implies\ &v > p_c^2. \end{aligned} \tag{23}$$

3.1.3 Debt asset price increase

Alternatively, if a price update raises the value of their debt at time t_1 by a factor of $p_d \in [1, \infty)$, then $d(t_1) = p_d c(t_0)v$, and their health is now

$$h(t_1) = \frac{c(t_1)}{d(t_1)} \cdot v = \frac{c(t_0)}{p_d c(t_0)v} \cdot v = \frac{1}{p_d}. \tag{24}$$

The user's health score has declined and the bonus percentage available to themselves for a self-liquidation is

$$b(t_1) = 1 - h(t_1) = 1 - \frac{1}{p_d} = \frac{p_d - 1}{p_d}. \tag{25}$$

As before, the malicious actor can therefore self-liquidate, receiving the entire collateral amount to their wallet. This total amount comprises a portion

$$c_b(t_1) = b(t_1)c(t_0) = \left(1 - \frac{1}{p_d}\right) c(t_0), \tag{26}$$

which is their liquidation bonus, and a remaining portion for repaying debt of

$$c_r(t_1) = c(t_0) - \left(1 - \frac{1}{p_d}\right) c(t_0) = \frac{1}{p_d} c(t_0), \tag{27}$$

which is the amount set aside for repaying of the borrowed asset. In this scenario, the price of collateral has not changed, but the value of the debt has increased, meaning the amount of debt required to be repaid for $c_r(t_1)$ units of collateral is decreased to

$$d_r(t_1) = \frac{c_r(t_0)}{p_d} = \frac{c(t_0)}{p_d^2}. \tag{28}$$

After repaying their flash loan using the collateral, and the ordinary loan using the assets they borrowed, there may be remaining debt on the protocol. This is the same value as the attacker's profit, which is given by

$$w(t_1) = d_r(t_1) - r(t_1) = c(t_0)v - \frac{c(t_0)}{p_d^2} = c(t_0)\left(v - \frac{1}{p_d^2}\right). \tag{29}$$

The malicious actor therefore profits from this sandwich attack when $w(t_1) > 0$, which implies

$$\begin{aligned} c(t_0)\left(v - \frac{1}{p_d^2}\right) &> 0 \\ \implies v &> \frac{1}{p_d^2}. \end{aligned} \tag{30}$$

3.1.4 Summary

For a flashloan attack, the attacker sandwiches a price oracle update, either when collateral decreases in price or the debt asset increases in price. Their goal is to flashloan and set up an unhealthy position that they can self-liquidate for profit moments later.

In both cases, the lower the LTV parameter v, the less likely an attack is to happen, because a much larger drop in price of collateral or much larger increase in the price of the debt asset is required for a malicious actor to be profitable.

Flashloan attacks cannot be made economically infeasible in the absence of other mechanisms without setting $v = 0$. They can only be made less likely by choosing v to be a value that is unlikely to ever correspond to a collateral price drop or debt asset price rise of a certain size.

It should also be noted that the severity of this attack can be amplified in extreme scenarios in which the oracle price update is so large that the malicious user's health score drops so much that it can be combined with the partial liquidation optimisation discussed above by ChainSecurity.

3.2 Multi-block attack

Whilst the attack described above relies on the use of a flashloan to exploit a price jump, taking advantage of a rare circumstance to guarantee a risk-free profit, a similar kind of attack can be carried out, in principle, across multiple blocks. This multi-block form of attack is much riskier from the attacker's perspective.

In a multi-block attack, a malicious actor sets up a position that is at risk of liquidation on block n and takes advantage of the price update with self-liquidation on block $n + 1$. This is much harder for the attacker for a variety of reasons.

First, they need up-front capital on block n to fund the attack. In the event of collateral price decrease they are choosing to hold onto this collateral to perform the attack even as the price is falling rapidly.

Second, they need to be sure that they can carry out self-liquidation on block $n + 1$. If they are fortunate enough to also be the proposer of block $n + 1$ this can be done without risk. However, if they are not, then they risk being liquidated by another liquidator at great cost to themselves. Given that the liquidation yields significant profit by design, competition among MEV bots is likely to be high, posing a significant risk to the malicious actor.

3.3 Oracle price jumps

It should be noted that sandwich attacks of this form are not unique to Euler's Dutch liquidation mechanism. They are possible on any lending protocols with fixed liquidation bonuses too. However, the Euler Vault Kit is potentially more vulnerable to sandwich attacks because it is designed to support pull-based oracles such as those from RedStone and Pyth, will sometimes report large jumps in price. These oracles pose greater risks for this kind of attack for two reasons.

First, pull-based oracles allow price oracle updates to be performed by users themselves, meaning that updates can be more readily included in a single sandwich transaction constructed by the attacker.

Second, pull-based oracles are not updated according to a heartbeat or threshold price, as with push-based oracles such as those provided by Chainlink, or with active trades, such as those provided by Uniswap TWAP oracles, but only when users choose to update them. For popular pull-based oracles, updates will tend to happen regularly and with small price deviations, but for newer or less popular oracles, there may be delays in updates which lead to large price jumps. It is these large price jumps that increase the risks of sandwich attacks.

3.4 Mitigation approaches

One way to potentially mitigate against the flashloan form of this attack is to require that liquidations can only happen sometime *after* the last passing health check on account. The principle here is that a loan originated or modified on block n that passed health checks should not become liquidatable on that very same block. In the extreme circumstance in which prices really did deviate on the same block, damaging a user's health, the user will just be eligible for liquidation on the next block rather than the current one, which is unlikely to pose too much risk to lenders. Including this kind of mechanism would rule out the flashloan variation of the attack, in which a flashloan is used to create a position whose *only* purpose is to be liquidated on the same block.

Blocking flashloan sandwich attacks would not, however, remove all risks. In particular, a multi-block version of this attack could potentially still be carried out by a well-capitalised malicious actor who is able to wait for just the right conditions to carry out the attack. As noted above, however, such a multi-block type of attack is likely to be considerably harder and riskier for a malicious actor, because they could lose a significant amount of money if things go wrong. In the event they did still try to carry out the attack, and it failed from their perspective, losing them money, it is important to note that the cost would still be borne by the lenders, with the benefit going to whichever third-party was able to receive the liquidation bonus created.

Another way to help limit the potential for the multi-block form of the attack is therefore to ensure that loans originated or modified by users always have a health score that exceeds 1. The further dis-

tance between the health score and 1 at the time a loan is originated or modified, the greater a price jump needs to be to make sandwich attacks possible. This can be enforced in practice either by stipulating different pricing mechanisms are used for liquidation vs loan origination/modification or by stipulating that different LTV parameters are used for liquidation vs loan origination/modification. In both cases, the goal would be to use more conservative pricing/LTV systems for a loan that is originated or modified than is used for liquidation. Euler supports both options.

3.5 Conclusion

There is no way to categorically prevent all classes of liquidation sandwich attacks. Many lending protocols are vulnerable to them to varying degrees. Flashloan versions of the attack can be prevented by disallowing liquidations that happen very soon after the origination or modification of a loan.

For multi-block versions of the attack, risk can only ever be reduced through prudent choices of oracles, assets, and lending protocol parameters, but never eliminated. The Euler Price Oracle currently allows vault creators to use more conservative pricing for loan origination/modification. The Euler Vault Kit allows vault creators to use more conservative LTV parameters for loan origination/modification.

Ultimately, the risks of these attacks are not unique to Euler. They are, however, elevated for vaults that rely on pull-based oracles, especially less popular price oracles that are not regularly updated. Vault creators and end users should carefully consider these oracle risks when creating/using vaults.

4 Omniscia

Omniscia proposes a interesting variant of the Dutch liquidation mechanism that is designed to prevent the liquidation bonus from being able to generate bad debt for a position that is solvent (i.e. over-collateralised).

Instead of having a liquidation incentive that starts low and rises as health drops below 1 (as in the design outlined above), they turn the mechanism on its head, and instead propose to have an incentive that starts high and falls. Specifically they propose that:

"*...liquidation incentives would be capped to the collateral value beyond the debt position's original value should the collateral's value exceed it. This approach would ensure that the liquidation incentives for unhealthy but adequately covered positions would be somewhat constrained to minimize / eliminate the bad debt that would be created otherwise. This would result in the liquidation incentive being high the moment a position turns unhealthy, to gradually lower as the position nears the toxicity threshold, and then to increase as the toxicity increases.*"

The liquidation bonus on offer to liquidators under the Omniscia proposal would be the percentage that gives liquidators all surplus collateral available. This can be expressed as:

$$b(t) = \frac{c(t) - d(t)}{d(t)} = \frac{c(t)}{d(t)} - 1 = \frac{h(t)}{v} - 1. \tag{31}$$

Note that in contrast to equation (3), where the bonus increases as health deteriorates, this equation gives rise to a bonus that decreases as health deteriorates. The bonus is zero when the position is no longer over-collateralised, which happens when $h(t) = v$. For a LTV parameter of $v = 0.8$, the

initial bonus on offer to liquidators would be $h(t) \approx 0.25$, or 25%. For $v = 0.95$, the initial bonus would be $h(t) \approx 0.0526$, or 5.26%.

This proposal would lead to very different liquidation dynamics than those expected under the mechanism described above. In particular, under the mechanism described above, positions are not necessarily expected to be liquidated as soon as they fall below a health score of 1, but only when the bonus becomes marginally profitable for liquidators. In contrast, the Omniscia proposal encourages swift elimination of positions the moment they fall below a health score of 1, because a high bonus leads to large profits for liquidators and encourages fierce competition among them to target positions as soon as they become eligible for liquidation.

The success of the Omniscia mechanism depends heavily on positions being liquidated in their entirety in a single liquidation. However, borrowers on vaults from the Euler Vault Kit are free to use multiple types of collateral to service loans. In the event that a borrower uses 2 or more collateral types, their position can only be partially liquidated in a single transaction. Under the Omniscia mechanism, the first partial liquidation would then lower, rather than increase, the liquidation bonus on offer for the next liquidator, moving the borrower closer to insolvency whilst simultaneously decreasing their chances of liquidation. Omniscia's proposal would therefore likely need to be accompanied by a significant change to the way that loans work on Euler v2, where borrowers would be constrained to use only one collateral at a time to service loans. For the sake of completeness, we will put the desirability of this constraint to one side for the time being and assume it is met for the remainder of the analysis.

One of the potential benefits of Omniscia's proposed mechanism is that positions that face liquidation are potentially liquidated swiftly without ever having the chance to generate bad debt. However, it should be noted that this might not always be the case. In the event that the maximum liquidation bonus is not sufficient to encourage swift liquidation of a position, the bonus would then fall as a position falls less and less healthy. In the event that prices or liquidity moves too quickly for liquidators to act or block congestion prevents liquidators from having their transactions included, then positions may enter a 'dead zone', where the bonus on offer has fallen below the threshold for profitability. Omniscia proposes that the bonus should rise again quickly when a position enters insolvency, but not until all surplus collateral has been offered to liquidators. In these scenarios it is plausible that the Omniscia mechanism could lead to delays in liquidating positions. With these delays, there might be more bad debt generation through their mechanism than the mechanism described above.

There are other downsides to the Omniscia mechanism too. As they note, such a high bonus would be bad for borrowers, encouraging them to avoid hard liquidation at all costs, potentially by self-liquidating themselves:

"*Whenever a position's liquidation would not result in bad debt, it is in the best interest of the borrower to liquidate themselves.*"

Whilst the incentive to self-liquidate is clear, there are many borrowers who will be unwilling or unable to do this. This is evidenced by the large number of borrowers on Compound, Aave, Maker, and others that have had large positions liquidated with high bonuses. Hard liquidations with large bonuses on these protocols have led to hundreds of millions in generated profit for MEV-type liquidation bots.

Whilst hard liquidation with big bonuses is bad for the borrowers themselves, it is also potentially bad for lenders, because every unit of collateral that is unnecessarily taken away from borrowers

decreases the likelihood that the remainder of their loans can be liquidated profitably without causing bad debt. This might not be a major risk when a position is liquidated in its entirety, but as noted above, borrowers on the Euler Vault Kit and other major lending protocols are generally able to use multiple collateral types at once to service loans, meaning partial liquidations are relatively common.

One counter-argument to the self-liquidation issue is that borrowers on vaults from the Euler Vault Kit are well-positioned to handle hard liquidations compared to other lending protocols because they can use Ethereum Vault Connector (EVC) 'operators' to define their own liquidation thresholds and dynamics through the use of mechanisms like stop-losses.

4.1 Conclusion

Whilst Omniscia's proposed alternative to the Dutch liquidation mechanism is interesting, it is one with significant trade-offs. Although it is designed to limit the production of bad debt on lending protocols, there are scenarios in which it might lead to *more* bad debt than alternative mechanisms. In that regard, it cannot be regarded as objectively better than alternatives. Vault creators who wish to implement such an alternative liquidation mechanism could build their own vaults and connect them to Euler's using the Ethereum Vault Connector.

5 Disclaimer

This post is for general information purposes only. It does not constitute investment advice or a recommendation or solicitation to buy or sell any investment and should not be used in the evaluation of the merits of making any investment decision. It should not be relied upon for accounting, legal or tax advice or investment recommendations. This post reflects the current opinions of the authors and is not made on behalf of Euler Labs or its affiliates and does not necessarily reflect the opinions of Euler Labs, its affiliates or individuals associated with Euler Labs. The opinions reflected herein are subject to change without being updated.