The Wildcat Protocol: Banking, But Worse

Laurence E. Day, Evgeny Gaevoy & Dillon Kellar

Whitepaper v0.2

September 4, 2023

Abstract

In this paper we present the Wildcat protocol, a protocol that permits ratified borrowers to establish fixed-rate, on-chain credit facilities, the collateral of which can be partially withdrawn for the borrower's purposes. It's different to what you've seen before - it's more flexible and far more general than the current solutions out there.

Wildcat is primarily aimed at organisations such as market makers wishing to borrow funds in the medium-term, but can reasonably be extended to parties such as DeFi protocols wishing to raise funds without the consequences of selling a native-token filled treasury into the ground to do so.

As a protocol that is - in its current form - reliant upon counterparty selection by borrowers, Wildcat is fundamentally permissioned in nature. Given that the positions that Wildcat vaults allow a lender to enter are undercollateralised by design, it is not suitable for usage in many cases, dependent on risk appetite.

Here we describe - in brief - our justification for creating Wildcat, a high-level technical sketch of the implementation (the full technical specification is a separate document), the lifecycle of a vault and handling of its various conditions, an overview of the legal protections for such vaults, and a faded map of directions for future work we will be building out after the prototype Ethereum mainnet release.

No, you're *probably* not the target audience.

No, there isn't a token right now.

Please stop asking for an airdrop.

Contents

1 Introduction

1.1 Motivation

This is a whitepaper about on-chain undercollateralised lending. And yes, we can already see your eyes rolling to the back of your head - DeFi and crypto writ large doesn't 'suit' the field well due to the pseudonymous, Sybil-vulnerable nature of it all - we rightly laud Aave, Euler, Compound et al for their solution of overcollateralised cross-asset lending, but these products exist to enforce the "don't trust, verify" nature of the industry we operate in.

Sometimes, we need to trust.

There are several protocols in operation at the time of writing that enable users to lend their funds in an undercollateralised manner to third parties who utilise them either on- or off-chain, such as Maple, Clearpool, Goldfinch, TrueFi, Centrifuge, Atlendis and even Aave with its credit delegation. These protocols serve their users well, but we believe that they have gaps or constraints in their product offerings relating to the choice of underlying, the interest rate methodology, and their usage of middlemen. We believe Wildcat addresses these in a way that makes it a more appealing choice for borrowers.

Firstly, these protocols generally advertise the ability to borrow a small number of assets - typically stablecoins and `WETH` - as their deposit numeraires, likely down to the fact that these are the most liquid and easy to ramp on and off exchanges. This works well enough - and some protocols permit more exotic assets to be borrowed - however, we are increasingly seeing the rise of off-chain agreements between entities for activities such as a) the hedging of positions or b) the market-making of newly released assets (see the collapse of Three Arrows Capital and release of the Worldcoin token as recent examples).

The promise of DeFi is transparency regarding arrangements such as this, and existing platforms are - in our opinion - missing a trick by not facilitating and encouraging these agreements to appear on-chain, rather than waiting for disagreement or disaster to force their terms into the open. It isn't as if we aren't seeing courts willing to intervene where they have jurisdiction and cause in digital asset default cases.

Consider this completely imaginary entity we've made up: an upcoming market-maker without deep cash reserves that cannot strike a deal with a founding team of a newly released asset `X`, but nonetheless wishes to provide a spread for it.

Their solution at present is to borrow a *different* asset such as `USDC`, swap this deposit asset for `X` and then hope that their fees taken from market-making `X` exceeds the sum total of any slippage or fees when acquiring `X`, any adversarial price action of `X` against the originally swapped asset, and the interest that they have to repay against the borrowed principal. Enabling the functionality to borrow arbitrary assets such as `X` via the same mechanisms as popular, liquid ones is desirable on the basis of such capital efficiency alone.

Secondly, the annual percentage rate (APR) on these deposits - whatever their underlying asset - is typically decided by way of a utilisation-based rate curve, whereby the interest paid by the borrower increases as the withdrawn funds approach a vault's capacity: the maximum amount that they can borrow. The exponential rate that these curves often adopt towards the final quartile can lead to perverse incentives whereby borrowers seek to take out a loan of more than they require and rehypothecate the excess to 'reduce' repayable interest.

Notwithstanding the moral hazards here where borrowers often 'second-hop' this excess capital into other lending protocols - often farming protocol token incentives as a result -, utilisation-based APRs mean that there is little certainty up front for a borrower in terms of what to repay. Similarly, a lender who contributes their capital to a popular vault at a certain rate can easily find themselves surprised if the borrower repays most of the outstanding balance shortly thereafter, dropping a utilisation APR to a point where they could better utilise those funds elsewhere. Depositing assets into a vault with a limited set of borrowers doesn't give rise to a mechanism to correct capital availability based on demand - your assets are deposited there exclusively for them, and their usage isn't directly parameterised by market sentiment or activity. In our opinion, in situations such as this you should be getting paid a fixed rate whether they're being used or not, until such time as a borrower decides they no longer have need for the credit facility.

Finally, proactive control of approved lenders and vault parameters from the perspective of a borrower is scarce. Control of - and intervention in - vaults of existing solutions are powers often granted to an in-housed 'delegate' or third-party protocol, relying on tools such as those provided by entities such as Chainalysis (proximity to sanctions) or Credora (a notion of credit-worthiness). In our opinion, this involvement impinges on the freedom to transact in a situation where a sufficiently grave legal agreement is perfectly capable of constraining counterparties to working within the bounds of sanity.

We believe that the intermediary roles that several of these extant protocols take up (such as determining maximum amounts that can be borrowed or retaining the ability to freeze pools upon default) are deterring agreements from being formed wherein the counterparties would otherwise be content to use Ethereum or other such networks to record and track them.

1.2 Features of Wildcat

The pieces that are novel here fundamentally relate to the freedom of the borrower to dictate the terms of the vaults that they deploy; freedom that necessarily comes with less intervention from the protocol itself:

- Vaults can be created for *any* ERC20 that the borrower wishes to acquire,
- Borrowers can determine the reserve ratio of a vault on creation, the minimum percentage of deposits that cannot be withdrawn but still accrue interest. Note the implication here that vaults are uncollateralised by a borrower: collateral is rather sourced from the lenders themselves,
- The maximum capacity of a vault can be adjusted at will up or down by the borrower depending on their current need,
- Fixed yield rates are chosen at deployment, specified in an APR that compounds each time someone interacts with the vault in a successful non-static call. Rates can be adjusted upwards by the borrower in order to incentivise lenders to participate (and downwards, subject to reserve ratio requirements),
- Borrowers determine the length of the withdrawal cycle: the rolling period during which multiple lenders will have their redemption amounts prorated if their sum exceeds the available vault collateral,
- Borrowers can choose a grace period after which vaults that have an insufficient reserve ratio incur a penalty APR that indicate the strength of a borrower's intention to maintain promised levels of collateral,
- Borrowers maintain their own list of addresses that are eligible to lend to the vaults deployed from a given controller contract, subject to their own KYC processes, and
- Borrowers can terminate a vault at any time, dropping the APR to 0% while simultaneously returning all outstanding collateral and interest: this blocks their ability to withdraw, and allows lenders to exit at their leisure.

From the perspective of lenders, their experience will look much the same as it currently does, with protections built-in via both a penalty rate for borrower delinquency and constraints on the borrower when dropping the APR rate, as well as an optional lending agreement that counterparties can choose to sign via ECDSA before engaging:

- Lenders can deposit to - and withdraw from - any active vault deployed by a borrower that has approved them on the relevant controller contract,
- Lenders that have had their approval to interact with a borrower revoked cannot deposit further, but retain the ability to withdraw,
- Lenders benefit from a penalty APR that triggers after a vault remains delinquent (under its reserve ratio) for a rolling length of time beyond the aforementioned grace period. This penalty is distributed to all lenders, and no part of it is receivable by the Wildcat protocol itself.

More generally:

Vaults do not have a minimum lockup period prior to redemption: lenders can request their deposit via entering a withdrawal cycle immediately after depositing should they choose, with the amount received as a result subject to potential proration given circumstance.

Depositing and redemption of vault debt tokens is restricted to those lenders permitted to interact with a given vault per the appropriate controller, however transfers beyond that are free game: you're welcome to LP them if you're a complete psychopath.

As alluded to above, these vaults can be governed by a lending agreement between a borrower and each of their lenders. Such agreements require unequivocal acceptance between lender and borrower of terms such as the jurisdiction in which to bring arbitration proceedings and/or civil claims against a borrower, and the conditions that bring about default.

The borrower is required to sign such an agreement (pre-populated with vault parameters) during the vault deployment process. This agreement is subsequently presented to any lender the borrower approves that attempts to engage with the vault through the protocol website. However, the lender is permitted to decline to countersign if they wish (perhaps they object to the obligations contained therein, or they have come to terms on an agreement that the borrower provides separately) - it is a template agreement we provide primarily to protect lenders from borrowers that do not have on-chain processes in place.

1.3 So, Who's This For?

As presented, Wildcat is 'just' a flexible credit facilitation protocol: if you wanted to use a term from traditional finance, it enables liquid fixed-income markets with a few twists. In its present form, it doesn't make use of oracles (since borrowers are explicitly borrowing and repaying the same asset that is loaned to them by lenders), and the assets that you - as a lender - deposit into a vault are interest-bearing under the *very important* proviso that the borrower ultimately redeposits them so that you can lay claim to your notional plus interest.

So who's sitting up and taking notice of this? We envisage a handful of cases:

- As suggested a couple of times now, market makers as borrowers who wish to acquire a supply of a newly released token in order to facilitate trading, either via liquidity provision or operating in a central limit order book (and it might not be a surprise to you that the protocol was initially developed for exactly this purpose).
- Legal entities attached to DAOs who wish to raise funds for development/operational purposes that have most of their treasury value denom-

inated in their native token and cannot (or will not) sell said token for assets that are more commonly accepted due to the potential price impact,

- DAO treasuries that have desirable - but nonproductive - assets in more diversified holdings that wish to utilise them in a yield-bearing manner by providing credit facilities to borrowers,
- Similarly to the above, ‘retail’ lenders (you reading this, family offices and so on) that have nonproductive assets without native yield-bearing variants that they are willing to utilise in the form of credit to a borrower provided they are willing and able to pass their KYC checks,
- Off-chain entities as borrowers that wish to raise funds on-chain that do not - or cannot - interface with the traditional banking system or institutional crypto platforms, and
- Third-party protocols that may be interested in listing vault tokens issued in the name of a borrower whereby they trade at a discount or premium reflecting market confidence in the borrowers ability to repay their loan.

It is not our place to circumscribe Wildcat’s usage: the above is simply a list of those parties that could *potentially* be served well by adopting it.

We do need to re-emphasise here that counterparty risk is very real in all cases: a borrower that defaults, collapses or disappears into the void after launching a vault and taking on debt can be sued in a court, but you’re unlikely to see a full return of assets if they’ve gone and done an Alameda. We encourage lenders to be sensible users of the platform - if you observe that a borrower has created a 100 million `WETH` vault with a 100% APR and a 1% reserve ratio, you might want to consider if the borrower deployer address has been compromised.

Now, let’s look at how this actually works under the hood.

2 Protocol Details

2.1 High-Level Overview

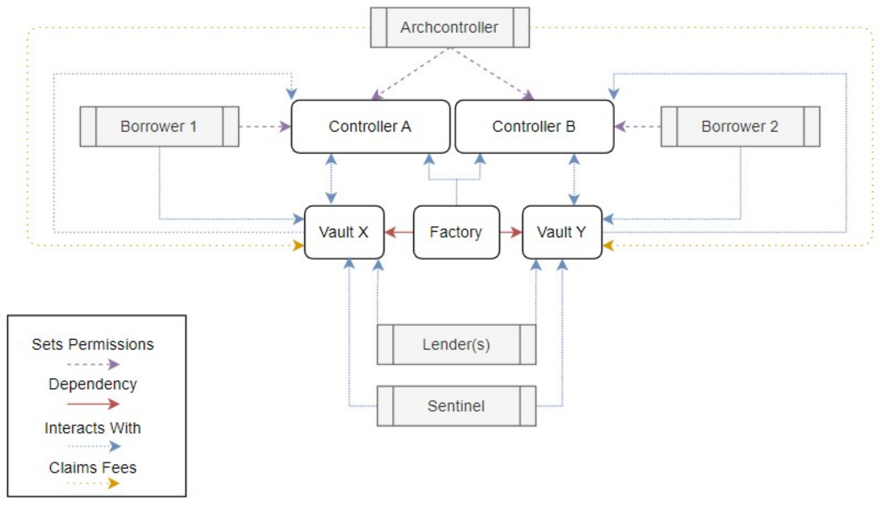
There are three main components to the Wildcat protocol: the *controllers*, the *factory*, and the *vault tokens* (or rather, the *vaults*):

- Controllers handle permissions for both borrowers and lenders alike,
- The factory calls into a controller when deploying a vault,
- Vaults query a specified controller to determine who can interact.

The address that owns a given controller is referred to as an *archcontroller*. This address represents the entity responsible for dictating the borrower that can deploy vaults using said controller, and will typically be operated by the Wildcat protocol itself. Another Wildcat-operated multisig exists - the *sentinel* - that has restricted powers to shift the balance and debt of a lender to an auxiliary escrow contract in circumstances involving sanctions (which we will address later).

No part of the protocol makes use of any proxy upgrade patterns, rendering any given deployed vault immutable. To emphasise: Wildcat does not have the ability to upgrade vault logic *even in the event of an emergency*. With the exception of the sanctions-related powers of the sentinel, at no point can Wildcat access (or take custody of) any assets within vaults beyond the fees incurred by the borrower.

If you're a fan of diagrams, here's one for you:



A controller contains:

- The address the archcontroller permits to deploy vaults (the borrower),
- The mechanism through which borrowers permit their lenders to interact,
- The sanity-check logic for vault deployment parameters, and
- Mechanisms for vault parameter control (e.g. APR adjustment)

Several controllers can exist concurrently, with different rules for permitting vault access and manipulating parameters. In the current version of the protocol, there is only one *variant* of controller.

The factory contains the initcode of a template vault, verifies (and/or rejects) the parameters to a vault provided by a borrower, and handles instantiated vault deployment via some neat `CREATE2` logic that we won't go into here.

The vaults are the important bit, so they get a few pages all to themselves.

2.2 Day-To-Day Usage: Your Vault And You

In this section, we'll explain how Wildcat vaults work by way of a long-running example: that of a borrower - we'll call them West Ham Capital - creating a vault in order to take out up to a thousand WETH on credit.

A vault is deployed with the following parameters:

- Asset Address: `0xC02aaA39b223FE8D0A0e5C4F27eAD9083C756Cc2`
- Symbol Prefix: `whc`
- Name Prefix: `West Ham Capital`
- Maximum Capacity: 1,200
- Reserve Ratio: 10%
- Vault APR: 5%
- Penalty APR: 3%
- Grace Period: 5 days
- Withdrawal Cycle: 7 days
- Sentinel Address: `(As Deployed)`
- Controller Address: `(As Deployed)`
- Protocol Fee: 20%
- Fee Recipient Address: `(As Specified By Controller)`

The result is a newly created contract that receives `WETH` and issues `whcWETH` (the full vault token name is West Ham Capital Wrapped Ether). Assuming that the borrower has previously instructed the specified controller to permit lenders to interact with their vaults (having put them through their own internal process to determine their suitability), deposits can now be accepted.

You might be thinking that there's a typo in the above, and that the capacity of the vault is too high compared to the desired amount to borrow (1,200 `WETH` versus a desired 1,000). That's intentional, given the 10% reserve ratio:

- A vault with a capacity of 1,000 could only withdraw 900 maximum,

- Similarly, a vault that set its capacity to 1,111 to withdraw up to 999.9 is likely to find itself delinquent nearly immediately.
- 1,200 allows for 1,080 to be withdrawn, but if the borrower constrains themselves to a maximum of 1,000, there is at least some buffer space beyond the required reserves for lenders to withdraw interest.

The 'best' choices here involve trade-offs between a number of factors that any given borrower must consider - a higher reserve ratio means there is more 'dead capital' they are paying interest on and requires a larger capacity, but reduces the maximum loss for lenders on aggregate in the event of a default. Similarly, a low grace period and short withdrawal cycle paired with a high penalty APR infers a strong commitment to maintaining redemption liquidity, but will likely require a far more hands-on approach to vault management by the borrower.

In the 'boring' case where everyone co-operates and no parameters are changed, a maximum of 1,200 `WETH` is deposited immediately and the outstanding supply inflates at a rate of 6% APR to 1,272 `WETH` after a year.

"Wait, that APR is wrong too, you said it was 5%!". Yes and no - the protocol fee of the vault was set at 20%: this is the *extra proportion of the vault APR* that routes to the Wildcat protocol - we have to make revenue *somehow*. In the front-end, the APR a lender earns is presented net of this fee (they'll see 5%): the protocol fee is for the borrower to concern themselves with.

One final point to add here is that the reserve ratio applies to the *vault supply*: if no lender withdraws any of their position during the year, the borrower can then withdraw a maximum of 1,144.8 `WETH` (an extension of 90% of the 72 `WETH` interest accrued).

But you're not interested in reading about peace. You want problems.

So what happens in the scenarios where things *change*? Let's cover them:

2.2.1 Borrower Adjusts Capacity

If a borrower decides that they want more capacity in their vault, they're free to change it at will. Increasing the capacity to 2,000 `WETH` results in a minimum liquidity reserve of 200 *when the vault is at its new capacity*: the change will not immediately result in delinquency.

If a borrower decides they want less, this is also fine - dropping the capacity does not have any impact on assets that are currently in the vault, as the capacity dictates openness to *new* deposits. If a lender deposits 100 `WETH` into a vault and the borrower decides that actually, that'll do, they can drop the maximum capacity to zero if they wish, but they're still on the hook to ultimately repay the lender that 100 `WETH` plus any interest accrued.

2.2.2 Borrower Adjusts Vault APR

A borrower that decides to increase the vault rate to induce their approved lenders to deposit (as approving a lender to interface does *not* compel them to) is able to do so at will. If the borrower of the aforementioned vault decided to increase the vault APR from 5% to 9%, their total rate of borrowing would become 10.8% when factoring in the protocol fee.

A borrower that *decreases* the vault rate will find themselves potentially having to return assets to the vault in order to meet a *temporarily* higher vault reserve ratio - the precise amount depends on the controller that deployed said vault - in the same transaction. Not requiring this would silently enable a rugpull mechanic whereby a borrower could induce deposits at a high rate, then withdraw as much collateral as the reserve ratio permits and drop the rate to reduce their future interest obligation.

We're not interested in permitting such behaviour, and we'd rather give the benefit of flexibility to lenders that decide that the new rate is no longer acceptable to them. We're aware that the relationship between APR and credit is not linear - in that a 20% decrease in APR may induce far more than 20% of deposited assets to exit - but by way of *example*, *if* the reserve ratio required when decreasing the APR is the larger of the existing ratio or the absolute value of the relative change between rates:

$$NewReserve = maximum\left(\left|\frac{NewRate \; - \; OldRate}{OldRate}\right|, \; OldReserve\right)$$

- then dropping the vault APR from 5% to 3% in a brand new, fully subscribed 1,200 `WETH` capacity vault with a 10% reserve ratio would temporarily set the reserve ratio to 40%, requiring:

- 0 `WETH` to be returned if all assets are still within the vault, and
- 360 `WETH` to be returned if assets have been withdrawn to capacity.

As a point of etiquette, we ask borrowers to inform their lenders one withdrawal cycle in advance of their intention to reduce the vault APR. If a lender has countersigned the underlying legal agreement before entering the vault, etiquette becomes covenant, and the failure of a borrower to do so is an event of default.

2.2.3 Borrower Allows Vault To Become Delinquent

In the event that a lender chooses to withdraw their assets (recall their loan) while the borrower has borrowed from the vault up to its capacity, it's likely that after the lender has withdrawn the vault will be under its reserve ratio. Using our aforementioned example:

1. Two lenders deposit into the vault: Alice lends 1,050 `WETH`, Bob lends 150,

2. Borrower withdraws 1,000 `WETH`, leaving 200 `WETH` in the vault,
3. Lender Bob decides to fully exit, and queues a withdraw of 150 `WETH` plus interest accrued (burning all their `whcWETH` tokens to do so),
4. The 150 `WETH` plus interest is added to a *reserved assets* pool in the vault which tracks honoured withdrawals that have not yet been claimed by their lender/s,
5. The vault now has just shy of 50 `WETH` in active reserves.

Interest ceases to be paid on Bob's deposit from the moment that the `whcWETH` tokens were burned, regardless of the length of the withdrawal cycle.

Given that the 10% reserve ratio of the vault requires that it contains at least 105 `WETH` in reserves (10% of the remaining 1,050 `WETH` supply from Alice), this vault immediately becomes delinquent. In this example, the borrower is required to return slightly over 55 `WETH` to 'cure' the delinquency.

Once a vault becomes delinquent, the grace tracker kicks in - a vault-specific rolling length of time beyond which the penalty APR activates. This is an additional rate accruing to remaining lenders until the grace tracker *drops back below the specified grace period*. To illustrate with our example vault:

1. The vault (which has a 5 day grace period) becomes delinquent,
2. The grace tracker begins to increase, however the base rate remains 6% for now (5% vault, 1% protocol),
3. 8 days pass before the borrower restores the reserve ratio, after which the vault ceases to be delinquent and the grace tracker begins to decrease,
4. A total of 6 days of penalty APR are eventually applied - the 3 days beyond the grace period before the vault was 'cured', and another 3 days to drop the grace tracker below the grace period,
5. The APR calculated for these 6 days is 9% (6% base, 3% penalty).

The protocol fee does not factor in the penalty APR while the latter is active.

2.2.4 Lenders Withdraw Beyond Vault Reserves

It may be the case that one or more lenders attempt to exit the vault for an amount greater than the current reserves in the same withdrawal cycle. In this instance, we make use of a novel mechanism for recording pending and honoured withdrawal amounts. Consider the following:

- Our example vault has one lender who lends 1,200 `WETH`,
- Borrower withdraws 1,000 `WETH`, leaving 200 `WETH` in the vault,

- Lender initiates a withdrawal cycle, requesting 300 `WETH`. Since there is only 200 `WETH` left in the vault, they burn sufficient `whcWETH` to commit that 200 `WETH` to the reserved assets pool, and the remaining 100 `WETH` is marked as *pending*,
- The result of the cycle depends on the actions of the borrower:
  - If the borrower deposits 100 `WETH`, lender can fully withdraw after burning the additional `whcWETH` tokens needed,
  - If the borrower does nothing, lender can withdraw the 200 `WETH` from the reserved assets pool that they burned `whcWETH` for.

In the event that lenders cannot withdraw the full amount that was demanded in a given withdrawal cycle, all pending requests are batched together, marked as 'expired' and placed into a queue. Expired batches continue to accrue interest until sufficient reserves are within the vault to fully pay them off. Future withdrawal requests can only be honoured to the degree that there are reserves in the vault in excess of what is needed to fill the reserved assets pool.

The existence of pending withdrawals in a vault also impacts its reserve ratio - pending withdrawals are required to be 100% collateralised, with the overall reserve ratio applying to the remainder. Consider this example:

- A vault with an maxed out capacity of 1,000 `WETH` has a 20% reserve ratio, and the borrower has withdrawn 800 `WETH` (leaving 200 as required),
- A lender requests a return of 450 `WETH`, burning sufficient vault tokens to move the 200 `WETH` into the reserved assets pool, with 250 marked as pending,
- The supply of the vault is reduced to 800 `WETH` (since 200 now belongs to the lender to withdraw),
- The temporary collateral requirement for the vault while the pending withdrawal exists is now (250 * 1) + (550 * 0.2) = 360 `WETH`, leading to a temporary reserve ratio of 45% on the outstanding supply of 800.

One final point to make is that in the event that multiple lenders attempt a withdrawal of assets exceeding the reserves in a vault in the same cycle, the amount that can be claimed by each is measured *pro rata* compared to each of their claims. As a final illustration:

- A vault contains 200 `WETH` in reserves.
- Lender Alice requests a withdrawal of 300 `WETH`. They burn sufficient `whcWETH` to move the 200 `WETH` into the reserved assets pool, and the remaining 100 is marked as pending.

- Lender Bob subsequently requests a withdrawal of 100 `WETH`. Since there are no more assets in the vault that are not part of the reserved assets pool, his request is immediately marked as pending, and he is not asked to burn any `whcWETH` to do so.
- At the end of the withdrawal cycle, Alice will be permitted to withdraw *150* `WETH` (rather than the full 200 she reserved), and Bob will be permitted to burn sufficient `whcWETH` to withdraw the remaining 50 `WETH`. This is a 3-1 ratio of the 200 `WETH` in the reserved assets pool, and corresponds to the ratios of their total requested withdrawal.
- The portion of Alice's `whcWETH` that was burned but corresponded to `WETH` that Bob was permitted to burn `whcWETH` to withdraw instead is recorded internally. When the vault has sufficient reserves to honour her remaining 150 withdrawal, she will only be asked to burn `whcWETH` equivalent to the 100 `WETH` in her pending withdrawal.

Note that the borrower doesn't *have* to wait until a withdrawal cycle is taking place in order to deposit assets back into the vault - they are free to do so at any time, and we imagine that the majority of the discussion regarding this will take place between lenders and the borrower off-chain.

It is worth closing this section by pointing out that protocol fees are considered senior to withdrawal requests from lenders. Throughout this section we have presumed that vaults would be completely emptied of assets when withdrawing more than reserves - in practice, the maximum amount permitted to be withdrawn is the available reserves less any fees owed to the vault archcontroller.

2.2.5 Borrower Removes Access From An Active Lender

As we have seen, each borrower maintains a list of approved lenders specific to them on a per-controller basis, and these lenders can interact with any vaults that borrower deploys using that controller. This approval includes the ability to deposit and withdraw at will, up to capacity and down to their deposited assets plus interest accrued.

In the event that a borrower removes a lender from this list, they are - unsurprisingly - barred from depositing further to any of their vaults. However, in the event that a lender has an existing active position in a vault, they are permitted to withdraw at their leisure: they cannot be forced out.

2.2.6 Borrower Wishes To Terminate The Vault

At all times, a borrower reserves the right to terminate a vault. This is an edge case of the borrower electing to reduce the vault APR - the vault APR is set to 0% and the borrower is required to make a full return of all outstanding assets plus interest accrued up until that block in the same transaction.

After doing this, the ability for a borrower to withdraw assets is *permanently* frozen, and no further changes to *any* vault parameters can be made. It is then up to the lenders to withdraw their assets - as with all vault APR shifts, the borrower will be expected to inform their lenders in advance of their decision to terminate a vault.

2.2.7 Borrower Loses Their Private Key

There's not really much we can do here. We don't want to enable the ability to update the borrower address for a vault - if we *could* do this, someone with control of the appropriate archcontroller could set the borrower of a fully-subscribed vault to a wallet they control, withdraw assets up to the reserve ratio, and disappear.

If a borrower *does* lose their key, the ideal result is that lenders withdraw the vault reserves and come to terms with the borrower off-chain for the return of the outstanding assets from a different address via a collateral agreement that the vault is considered terminated (since the borrower can't do this themselves).

2.2.8 Lenders Lose Their Private Keys

This one is probably just tough luck - we're even less keen on the idea of introducing the ability for the protocol to move lender balances to arbitrary addresses than we are on being able to switching the borrower address, for the same reasons as in the previous section.

There is the *potential* to introduce a backup list of addresses for each lender along with the ability for a borrower to burn and re-mint vault tokens between lenders and their backups, but this strikes us as an immediate exploit vector, and would probably be the most vulnerable component of the entire protocol.

Far better - in both the lender and borrower cases - is to insist that any addresses that will touch a vault have built-in failsafes (e.g. Gnosis safes, key sharding).

2.2.9 Lender Gets Placed On A Sanctions List

There is *one* circumstance whereby the Wildcat protocol can alter the active balance of a lender within a vault, and that is when their address is added to a collection such as the OFAC SDN List or the UK Sanctions List.

We have previously mentioned the *sentinel* address. This address is tied to a service that periodically verifies that none of the addresses with an active position in any Wildcat vaults are present on such lists that can be checked on-chain.

In the event that a lender is flagged in such a way, the sentinel can *elect* to call a 'excision' function - which will require that the target address in question is sanctioned according to the Chainalysis sanctions oracle contract before triggering - within any affected vault. The effect of this would be that an auxiliary escrow contract is deployed and the debt obligation of the borrower towards the lender is transferred to this new contract. Interest does not accrue within this contract from the time the debt is transferred, and the borrower is expected to immediately return the balance of funds due up to that time to the escrow contract. Funds can be retrieved by the lender from the contract either after such time that the address is removed from the appropriate sanctions list or if the the borrower receives a court order from the relevant jurisdiction.

Note that the sentinel address falling into hostile hands will have limited impact on a vault beyond potentially requiring said vault to migrate to a new deployment: the *only* thing it can do is move debts to factory deployed contracts which can subsequently be resolved peacefully between counterparties, as the lender would not be on a sanctions list. It is highly doubtful that the same party that seizes control of a sentinel can also manipulate Chainalysis.

2.2.10 Borrower Gets Placed On A Sanctions List

If the sentinel detects that a *borrower* is added to a sanctions list, sorry, but it's just going to have to go straight to court per the terms of the protocol service agreement. Reserves can be pulled from the vault by lenders, but subsequent return of assets would likely incur strict liability for dealing with a sanctioned party and poison the address of any lender that withdrew thereafter.

2.2.11 That's All She Wrote

So - that's how you utilise the Wildcat protocol. Any other vault parameter we haven't mentioned in an adjustable scenario cannot be changed when the vault is live, and the operators of the protocol itself do not custody or have control of your funds at any point.

3 Future Directions

With that said - there are a couple of things that don't exist in the current implementation of the Wildcat protocol that we want to get around to adding in, provided that our lawyers don't burst into laughter when we suggest them.

Here's a non-exhaustive list for an idea of what we're thinking about:

3.1 Long-Dated Option Vaults

One of the most common forms of agreements between market makers and the teams behind newly released tokens at present goes along these lines:

"Lend us X% of your token for a year - we'll market make with them in the appropriate venues as and when they become available - and grant us the right to purchase these tokens from you instead of having to return them at the end of the loan at a given strike price."

This is a long-dated option, and is perfectly viable as a type of agreement to build into a Wildcat vault. You don't even need to integrate an oracle for this provided that the strike isn't floating - it's a couple of additional parameters in a factory tied to an upgraded controller that permits a borrower to deploy a 0% APR vault and elect to change the underlying asset to the proposed alternative.

3.2 Credentialed Access

A borrower may not be confident in their ability to safely gauge a candidate lender's suitability, or wishes to avoid any data privacy/protection requirements that they may fall under by doing so. An alternative to maintaining a borrower-specific set of lenders is to defer the process entirely to a third-party.

Identity solutions such as Circle's Verite permit expiring credentials to be issued to a wallet, allowing a borrower to select for parameters such as nationality and accredited investor status by making use of a controller that integrates a verifier contract. This has the added benefit of opening the pool of potential lenders to everyone that has already onboarded to such a credential issuing service, rather than repeating the process themselves for the smaller set of candidates that they have established a relationship with.

We are interested in investigating the potential to allow - for example - all Circle account holders to automatically be permitted to deposit into vaults.

3.3 Interest In Different Tokens

At present, each vault is denominated in a single token: this simplifies calculations and allows us to avoid the usage of oracles (in our opinion the biggest exploit vector when it comes to vault-related protocols), but may give rise to issues for the borrower in the situation where they cannot acquire enough of a token to cover the interest due on top of whatever they have borrowed.

We are interested in exploring the potential of permitting a borrower to repay interest in the form of a *different* token - i.e. accepting `USDC` deposits but repaying interest in `DAI` (or, more exotically, a combination of the two).

A mechanism such as this would enable entities such as DAOs to pay interest on loans in a native token, but introducing this would both require oracle integration into Wildcat and the presence of a feed for said native token in order to establish the correct amounts to release. Interesting, but dangerous.

4 Code Repository

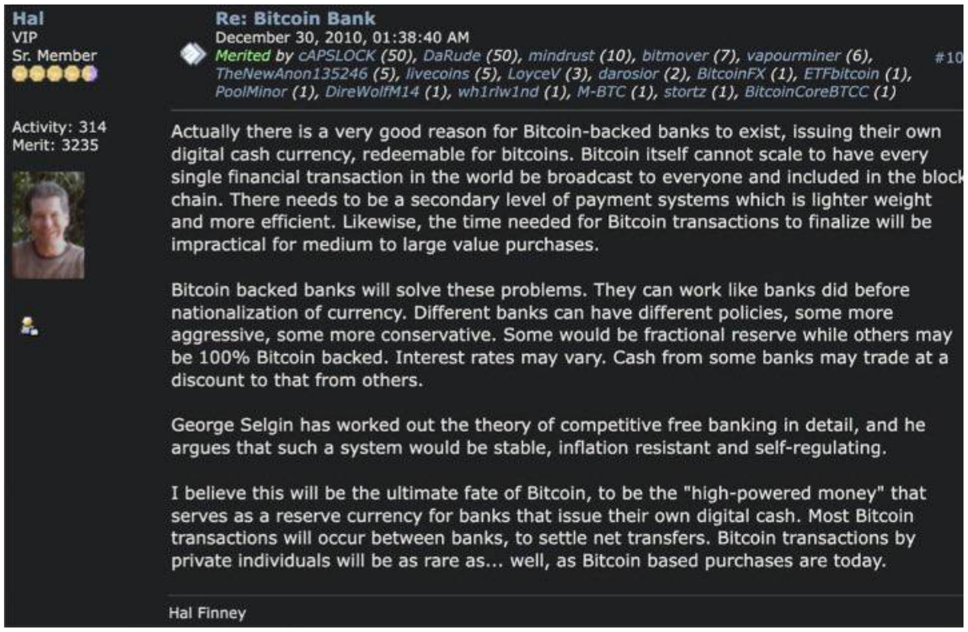
The code for the Wildcat protocol will be made available under the Commons Clause License, and can be accessed - along with the most recent version of this whitepaper and the technical specification - at:

`http://www.github.com/wildcat-finance`

If you're not familiar with this license, it means that you're welcome to use it at will in terms of building upon it and monetising any 'value-add' that you provide, however you cannot sell the *protocol code* itself. That belongs to us.

5 One Final Point

Hal Finney wanted this. He was just - as always - early.



Hal
VIP
Sr. Member

Activity: 314
Merit: 3235

Re: Bitcoin Bank
December 30, 2010, 01:38:40 AM
Merited by cAPSLOCK (50), DaRude (50), mindrust (10), bitmover (7), vapourminer (6), TheNewAnon135246 (5), livecoins (5), LoyceV (3), darosior (2), BitcoinFX (1), ETFbitcoin (1), PoolMinor (1), DireWolfM14 (1), wh1rlw1nd (1), M-BTC (1), stortz (1), BitcoinCoreBTCC (1)
#10

Actually there is a very good reason for Bitcoin-backed banks to exist, issuing their own digital cash currency, redeemable for bitcoins. Bitcoin itself cannot scale to have every single financial transaction in the world be broadcast to everyone and included in the block chain. There needs to be a secondary level of payment systems which is lighter weight and more efficient. Likewise, the time needed for Bitcoin transactions to finalize will be impractical for medium to large value purchases.

Bitcoin backed banks will solve these problems. They can work like banks did before nationalization of currency. Different banks can have different policies, some more aggressive, some more conservative. Some would be fractional reserve while others may be 100% Bitcoin backed. Interest rates may vary. Cash from some banks may trade at a discount to that from others.

George Selgin has worked out the theory of competitive free banking in detail, and he argues that such a system would be stable, inflation resistant and self-regulating.

I believe this will be the ultimate fate of Bitcoin, to be the "high-powered money" that serves as a reserve currency for banks that issue their own digital cash. Most Bitcoin transactions will occur between banks, to settle net transfers. Bitcoin transactions by private individuals will be as rare as... well, as Bitcoin based purchases are today.

Hal Finney

See you on-chain.

6 Changelog

The latest version of this whitepaper can be found at:

`https://github.com/wildcat-finance/wildcat-whitepaper`

- Version 0.2 [4 September 2023]
  - Emphasised that no upgrade powers are possessed by protocol
  - Introduced withdrawal mechanism
  - Softened the impact of the sentinel from deletion to excision
  - Highlighted that protocol fees are senior to lender claims
- Version 0.1 [17 May 2023]
  - Initial draft