The Wildcat Protocol: Banking, But Worse

Laurence E. Day, Evgeny Gaevoy & Dillon Kellar

Whitepaper v1.0

November 13, 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ratified borrowers to establish fixed-rate, on-chain credit facilities, the collateral of which can be partially withdrawn for the borrower's purposes.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 markets 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), a description of the lifecycle of a market and the various scenarios users can encounter, 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 the overall 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 mitigate 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 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 market 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 facility.

Finally, proactive borrower control of approved lenders and parameters such as interest rate and capacity is scarce. Control of - and intervention in - existing solutions are powers often granted to an in-housed 'delegate' or third-party protocol. In our opinion, this involvement impinges on the freedom to contract 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 markets that they deploy; freedom that necessarily comes with less intervention from the protocol itself:

- Markets can be created for *any* non-rebasing ERC20 the borrower wants,

- Borrowers can determine the reserve ratio of a market on creation: the percentage of deposits that cannot be borrowed but still accrue interest. Note the implication here that markets are uncollateralised by a borrower: collateral is rather sourced from the lenders themselves,
- The maximum capacity of a market can be adjusted at will upwards (with no upper limit) or downwards (to current debt levels) 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 a market 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 discussed herein),
- Borrowers determine the length of the withdrawal cycle: a period of time in which multiple lenders making withdrawal requests therein will have their redemption amounts prorated if their sum exceeds available reserves,
- Borrowers can choose a grace period after which markets 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 their markets deployed from a given controller contract, subject to their own KYC processes, and
- Borrowers can close a market at any time, dropping the APR to 0% while simultaneously returning all outstanding collateral and interest: this blocks their ability to borrow, 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 defences built-in via both a penalty rate for borrower delinquency and constraints on the borrower when dropping the APR, 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 market 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 market 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:

Markets do not have a minimum lockup period: lenders can request their deposit via entering a withdrawal cycle - the length of which is determined by the borrower - immediately after depositing should they choose, with the amount received as a result subject to potential proration given circumstance.

Issuance and redemption of market tokens (representing debt) is restricted to those lenders permitted to interact with a given market 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 markets *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.

If interacting with Wildcat through the protocol front-end, a borrower is required to sign such an agreement (pre-populated with market parameters) during the market deployment process. This agreement is subsequently presented to any lender the borrower approves that attempts to engage with the market 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.

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 market are interest-bearing under the *very important* proviso that the borrower 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 market 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.

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 market and taking on debt can be taken to 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` market with a 100% APR and a 1% reserve ratio, you might want to consider if the borrower's 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 four main components to the Wildcat protocol: the *archcontroller*, the *market controller factories*, the *market controllers* and the *markets* themselves:

- The archcontroller acts as a global registry of deployed contracts and provides access control for borrowers,
- Market controller factories are approved contracts that contain the bounds for the various parameters of markets ultimately deployed through their instances and general logic for a particular 'type' of market.

- Market controllers are factories themselves (for markets) owned by individual borrowers. They contain lists of authorised lenders as well as functionality for adjusting and interfacing with markets they deploy.
- Markets are the front-facing 'product' of the protocol - contracts tied to a rebasing ERC20 token representing claims on debt which are deposited into by lenders and borrowed from and repaid to by borrowers.

2.1.1 The Archcontroller

The address which deploys and owns an archcontroller contract represents the entity operating a particular deployment of the Wildcat protocol. The extent of the powers of this address within the archcontroller are as follows:

- Authorising and removing addresses as borrowers, permitting - or forbidding - them to deploy market controllers (and hence markets themselves) through market controller factories. Removed borrowers can continue to interact with any of their existing markets unabated.
- Registering and removing market controller factories which can be used by authorised borrowers (removal of a controller factory prevents any further controllers from being deployed, but leaves existing ones untouched),
- Removing market controllers, preventing any further markets from being deployed from them (but leaving existing ones untouched), and
- Removing markets, which does not hinder their operation in any way, but rather hides them from the SDK for the purposes of displaying on any front-end which utilises it.

In effect, the only *meaningful* power the archcontroller owner has is deciding who can be a borrower. It can 'cease operations' of their Wildcat protocol deployment by removing all registered controller factories and controllers, but can in no circumstance interfere with already deployed markets.

Later, we discuss the *sentinel*, a contract which possesses restricted powers to shift market-associated balances of a lender to auxiliary escrow contracts in circumstances involving sanctions.

No part of the protocol makes use of any proxy upgrade patterns, rendering any given deployed market immutable. To re-emphasise: no party has the ability to upgrade market logic *even in the event of an emergency*. With the exception of the limited powers of the sentinel, at no point can Wildcat access (or take custody of) any assets within a market beyond the fees incurred by the borrower.

2.1.2 Market Controller Factories

A market controller factory contains a series of constraints that ultimately pass through to any markets that 'descend' from them via a market controller deployed through it - constraints such as 'the lender APR of a market cannot exceed 100%' or 'the minimum penalty APR is 5%'.

They also contain a given protocol fee configuration, which determines what - if any - fees are payable to the protocol, in the form of a fixed origination fee in a specified asset, a percentage of the lender APR added to the debt of a market, neither, or both. Note that the archcontroller owner *can* change these values after deployment, but they *do not have retroactive effect* - markets deployed with a previous fee configuration are untouched. This power exists to account for fee recipient addresses that can no longer be accessed or origination fee assets that are rendered unusable.

Finally, market controller factories contain the initcode and hashes for the template contracts of the market controllers and markets they ultimately facilitate. This latter feature means that a market controller factory contains the 'shape' of a particular market: distinct market controller factories can produce markets with vastly different functionalities while remaining under the umbrella of the same protocol deployment.

2.1.3 Market Controllers

Any given market controller is owned by the borrower that deployed it through a market controller factory. They contain sets of addresses representing lenders which the borrower has permitted to engage with the markets that the market controller deploys in turn.

More generally, market controller contains functionality for enforcing parameter constraints inherited from its parent controller factory in the deployment of any market, as well as functionality for adjusting the parameters that can be mutated for its child markets (lender APR and maximum capacity) and closing markets which a borrower no longer has need of.

A market controller inherits the initcode of template markets from its parent market controller factory. We're quite proud of the novel `CREATE2` logic which we use to handle instantiated market deployment: if you're a Solidity developer, maybe go and have a look, but we won't go into it here.

2.1.4 Markets

The markets are the important bit as far as you're concerned, and we'd do them an injustice summarising them here. So, they get a few pages all to themselves.

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

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

A market is deployed from a market controller with the following parameters:

- Asset Address: `0xC02aaA39b223FE8D0A0e5C4F27eAD9083C756Cc2`
- Symbol Prefix: `whc`
- Name Prefix: `West Ham Capital`
- Maximum Capacity: 1,200 *market tokens* (introduced shortly)
- Reserve Ratio: 10% of current market token supply
- Lender APR: 5%
- Penalty APR: 3%
- Grace Period: 5 days
- Withdrawal Cycle: 7 days
- Sentinel Address: `(As Deployed)`
- Controller Address: `(As Deployed)`
- Protocol Fee: 20% of lender APR
- Fee Recipient Address: `(As Specified By Controller)`

The result is a newly created market that accepts `WETH` and issues the market token `whcWETH` in exchange (the market token name is West Ham Capital Wrapped Ether). If the borrower has previously authorised lenders via the market controller which deployed the market - 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 maximum capacity of the market is too high compared to the desired amount (1,200 versus the desired 1,000). Consider the 10% reserve ratio:

- A market with a maximum capacity of 1,000 could only withdraw 900 `WETH` maximum, since 10% of the debt of a market must remain within the market as liquid reserves,
- A market that set its maximum capacity to 1,111 to withdraw up to 999.9 `WETH` maximum could find itself delinquent nearly immediately,

- By contrast, a maximum capacity of 1,200 allows for 1,080 `WETH` to be withdrawn if the market is fully subscribed. If the borrower constrains themselves to a maximum of 1,000 `WETH`, there is at least *some* buffer space beyond the required reserves for lenders to place withdrawal requests to cover their interest.

The 'best' choices for how to parameterise a market involve trade-offs between a number of factors that a borrower must consider. A higher reserve ratio means there is more 'dead capital' they are paying interest on and requires a larger capacity to reach their desired borrowable amount, but reduces the maximum loss for lenders on aggregate in the event of a default. In turn, 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 market 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 after market creation, 1,200 `whcWETH` market tokens are issued in exchange and the `whcWETH` supply 'inflates' at a rate of 5% APR to a total of 1,260 after a year. This 'inflation' is a result of the fact that market tokens issued in exchange for deposits are *rebasing*, so as to always be redeemable at parity based on the interest incurred by the borrower: 1 `whcWETH` represents a debt of 1 `WETH` owed by the borrower to the lender. The relationship between the underlying asset and the number of market tokens issued for them - as well as the growth of the latter - at any given point is governed by a *scale factor*, and we promise you'll thank us for not going into the details here.

Another specific figure worth highlighting here is that of the protocol fee, which is defined here as 20% of the 5% lender APR. This 'additional' 1% APR interest is claimable to the fee recipient address - we have to make revenue *somehow*! Note that the presence of a protocol fee is not guaranteed, and depends on the protocol fee configuration of the market controller factory from which the market is derived. Protocol fees do not factor into the rebasing rate of market tokens: they are tracked separately and claimed from liquid market reserves when requested.

One final point to add here is that the amount of reserves required within a market is a function of the market token supply, *not* the maximum capacity. In the above example, if you only receive 10 `WETH` in deposits, you will only need to maintain slightly over 1 `WETH` in reserves to meet the reserve ratio.

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 market, they're free to increase it at will. As referenced at the end of the previous section, increasing the maximum capacity to 2,000 `whcWETH` in our example only requires a minimum of 200 `WETH` in reserves *when the market is at its new capacity*: the change will not immediately result in delinquency.

A borrower is also free to reduce the capacity of a market, but only to the value of the current market token supply (i.e. the total amount of outstanding debt currently owed by the borrower).

2.2.2 Borrower Adjusts Lender APR

A borrower that decides to increase the lender APR to induce their authorised lenders to deposit is able to do so at will, up to the maximum permitted by the relevant controller. If the borrower of the aforementioned market decided to increase the lender APR from 5% to 9%, their total rate of borrowing would become 10.8% when factoring in the protocol fee and assuming non-delinquency.

A borrower that *decreases* the lender APR will find themselves potentially having to return assets to the market in order to meet a *temporarily* higher reserve ratio in the same transaction. Not requiring this would silently enable a rugpull mechanic whereby a borrower induces deposits at a high APR, withdraws as much collateral as the reserve ratio permits and drop the APR to reduce their future interest obligation.

We're not interested in permitting such behaviour, and we'd rather provide support to lenders that decide that a lower rate is no longer acceptable to them. The relationship between APR and willingness to grant credit is not linear - a 20% decrease in APR may induce *far* more than that amount of deposits to exit, but if there aren't enough assets in reserve, they're left to the mercy of the borrower. At the same time, however, insisting on a full return of assets for a small decrease (i.e. to maintain a certain rate above T-Bill yield for a stablecoin market) renders the credit facility useless to the borrower while still leaving them liable to pay interest - in that situation they might as well force a market closed and restart a different one. As a middle-ground, then, when reducing lender APR the reserve ratio required is set for two weeks as the larger of the *existing* ratio or double the relative change between rates (capped at 100%):

$$NewReserve = max(max(100\%, 2 * \frac{OldRate - NewRate}{OldRate}),\ OldReserve)$$

This means that reducing a market APR from 5% to 3% in a market with a 10% reserve ratio would temporarily set the reserve ratio to 80% of the current market token supply, regardless of the underlying asset.

This also means that the borrower that reduces lender APR by 50% or more in one shift will be required to fully collateralise the market for a fortnight. After this time, any address can *reset* the ratio back to its previous value provided that the market is not delinquent under the *higher* ratio when doing so.

2.2.3 Borrower Allows Market To Become Delinquent

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

1. Two lenders deposit into the market: Alice lends 1,050 `WETH`, Bob 150,
2. Borrower borrows 1,000 `WETH`, leaving 200 `WETH` of reserves in the market,
3. Shortly thereafter, lender Bob decides to fully exit, and queues a withdrawal, transferring all of their `whcWETH` tokens to the market to do so,
4. The market burns as many `whcWETH` as possible at a 1:1 rate in order to move liquid `WETH` reserves to an *unclaimed withdrawals pool* in the market which tracks withdrawals that have not yet been claimed by their owner,
5. Given that the market token balance of Bob would have been slightly over 150 `whcWETH` after rebasing to account for interest, the market now has just shy of 50 `WETH` in liquid reserves.

Interest ceases to be paid on Bob's deposit from the moment that all of his `whcWETH` tokens were burned, regardless of the length of the withdrawal cycle (we will discuss what happens if the withdrawal request amount exceeds the liquid reserves available shortly).

Given that the 10% reserve ratio of the market requires that it contains just over 105 `WETH` in reserves (10% of the remaining 1,050-and-change `whcWETH` supply still held by Alice), this market immediately becomes delinquent. In this example, the borrower must return slightly over 55 `WETH` to 'cure' the delinquency.

Once a market becomes delinquent, the grace tracker kicks in - a market-specific rolling length of time beyond which the penalty APR activates. This is an additional rate that increases the rebasing speed of market tokens via the scale factor (and thus accrues more interest to lenders) until the grace tracker *drops back below the specified grace period*. To illustrate with our example market:

1. The market (which has a 5 day grace period) becomes delinquent,
2. The grace tracker begins to increase, however the APR (reflecting the rate of growth of the scale factor) remains 5%,

3. 8 days pass before the borrower repays enough to meet the minimum reserves, after which the market 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 market was 'cured', and another 3 days until the market has been delinquent for less than the 5 days grace,
5. The APR calculated for these 6 days is 8% (5% base, 3% penalty).

The protocol fee does not factor in the penalty APR while the latter is active, remaining unchanged at 1% throughout.

2.2.4 Lenders Withdraw Beyond Market Reserves

It may be the case that one or more lenders place withdrawal requests 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 market has one lender who lends 1,200 `WETH`,
- Borrower withdraws 1,000 `WETH`, leaving 200 `WETH` in the market,
- Lender initiates a withdrawal cycle, requesting 300 `WETH` and transferring 300 `whcWETH` to the market to do so.
- The market burns 200 `whcWETH` to commit the reserves to the unclaimed withdrawals pool: the remaining 100 `whcWETH` is marked as pending,
- What happens after this depends on the actions of the borrower:
  - If the borrower deposits 100 `WETH`, lender can make a claim for the full 300 `WETH` at the end of the cycle after the market burns the pending 100 `whcWETH` market tokens to mark their request as fully honoured,
  - If the borrower returns nothing, the lender can claim 200 `WETH` at the end of the cycle, with the pending 100 `whcWETH` associated with their request remaining in the market.

In the event that lenders cannot claim the full amount that was demanded in a given withdrawal cycle, all pending requests are batched together, marked as 'expired' and placed into a FIFO queue. Expired batches continue to accrue interest until sufficient reserves are within the market to fully pay them off, since there are still market tokens associated with them. These market tokens will continue to rebase until they are burned while assigning later repayments to expired batches in the order that they were added to the queue.

The existence of a non-zero withdrawal queue in a market impacts its reserve ratio - the amount requested by the queue is required to be 100% collateralised, with the reserve ratio applying to the remainder. Consider this example:

- A market 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 makes a withdrawal request for 450 `WETH`: the market burns 200 `whcWETH` to move the 200 `WETH` into the unclaimed withdrawals pool, and leaves 250 `whcWETH` untouched,
- The supply of the market is reduced to 800 `whcWETH` (since 200 `whcWETH` was burned to move the reserves),
- The temporary collateral requirement for the market 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 `whcWETH` rather than the typical 20%.

One final point to make is that in the event that multiple lenders attempt a withdrawal of assets exceeding the reserves in a market 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 market contains 200 `WETH` in reserves.
- Lender Alice requests a withdrawal of 300 `WETH`. The market burns 200 `whcWETH` to move the 200 `WETH` into the unclaimed withdrawals 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 market that are not part of the unclaimed withdrawals pool, his request is immediately marked as pending, the market does not burn any of the 100 `whcWETH` he transferred.
- At the end of the withdrawal cycle, Alice will be permitted to claim *150* `WETH` (rather than the full 200 she reserved), and Bob will be permitted to claim the remaining 50 `WETH`. This is a 3-1 ratio of the 200 `WETH` in the unclaimed withdrawals pool - a ratio corresponding to their total requested withdrawal amounts.
- The withdrawal queue now contains an expired batch of 150 `whcWETH`. These tokens will continue to rebase, and any assets that are repaid to the market and assigned to this batch (burning the pending `whcWETH` in the process) are claimable pro-rata until the batch is fully honoured.

Note that the borrower doesn't *have* to wait until a withdrawal cycle is taking place in order to repay debt to the market - 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 any protocol fees which have accrued are considered senior to withdrawal requests from lenders. Throughout

this section we have presumed that markets would be completely emptied of assets when attempting to withdraw more than reserves. In practice, the maximum amount permitted to be withdrawn is the available reserves less any fees owed to the protocol.

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 markets that borrower deploys using that controller. This approval includes the ability to deposit and withdraw, subject to capacity and reserves.

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

2.2.6 Borrower Wishes To Close The Market

At all times, a borrower reserves the right to close a market. This is an edge case of the borrower electing to reduce the lender APR - the lender APR is set to 0% and the borrower must repay enough to the market to meet a 100% reserve ratio.

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

Note that in the event that a market is delinquent at the time that it is closed, the grace tracker is set to zero so as to not incur any further penalty interest. This may come off as an easy 'out' for a delinquent borrower since by rights they 'should' pay the penalty imposed until the grace tracker is back within range of the grace period, but our perspective is that lenders would sooner just be able to access their assets plus interest accrued up until the point of closure.

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 market - if we *could* do this, an attacker seizing control of the archcontroller owner address could set the borrower of a fully-subscribed market to a wallet they control, withdraw assets up to the reserve ratio, and disappear.

If a borrower *does* lose their key, it is possible for third-parties to transfer assets directly into the market contract and subsequently update the market state.

The market might not be able to be officially *closed* in this instance, but closure can happen for all intents and purposes here via the borrower transferring their debt from another address and all lenders placing full withdrawal requests.

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 archcontroller owner 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: an attacker could transfer market tokens to a wallet they control and then place withdrawal requests.

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 market 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. At present, it is not something we support.

Far better - in both the lender and borrower cases - is to insist that any addresses that will touch a market 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 market tokens for a lender within a market, and that is when they are added to a collection such as the OFAC SDN List or the UK Sanctions List.

We have previously mentioned the *sentinel*. This is a contract which is a) used to check that parties interacting with Wildcat markets are not sanctioned by the Chainalysis oracle which records addresses on these lists, and b) a factory for 'escrow contracts' which hold assets belonging to sanctioned lenders.

In the event that an lender is flagged as sanctioned via the sentinel, what happens thereafter depends on the market interaction. If they are attempting to deposit, their transaction will be reverted. If they are attempting to claim assets after a withdrawal request, the sentinel will create two escrow contracts - one to hold the underlying assets that would otherwise be transferred to the sanctioned address, and another to hold the balance of any market tokens they may still hold.

Within each market contract, there is also an excision function which can be invoked to transfer the market token balance of a sanctioned lender to an escrow contract, provided that the sentinel condemns them as sanctioned at the time of the function call.

Assets can be retrieved from escrow contracts in one of two circumstances:

- The lender address in question has its sanction lifted and the Chainalysis oracle is updated such that the sentinel no longer flags the address, or
- The borrower elects to *override* the sanction - an ability they can invoke directly through the sentinel contract itself. We expect that a borrower will not use this power without *very* good reason.

2.2.10 Borrower Gets Placed On A Sanctions List

If the sentinel detects that a *borrower* is added to a sanctions list, any attempt to borrow further from any market they deployed will revert. Beyond this, lenders are in a pickle: existing reserves can be withdrawn from the market by lenders, but subsequent repayment of assets by the borrower could potentially incur strict liability offences on the part of a lender.

We don't have a good answer present *in code* in this case, and would advise speaking to lawyers if this happens.

2.2.11 That's All She Wrote

So - that's how you utilise the Wildcat protocol. Any other market parameter we haven't mentioned in an adjustable scenario cannot be changed when the market is live, and the operator of the protocol deployment itself - the archcontroller owner - does 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 of things we're thinking about:

3.1 Long-Dated Option Markets

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 a perfectly viable candidate for a Wildcat market type. You don't even need to integrate an oracle for this provided that the strike isn't floating - it requires a market controller factory defined in a way that permits a borrower to deploy 0% APR markets which a) prevent withdrawal requests for a fixed period of time and b) permit the debt to be repaid via fixed amounts of an alternative underlying.

This is very likely the first protocol upgrade we release.

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 and Coinbase Verifications permit expiring attestations to be tied 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 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 of introducing - for example - a market controller which has the option of permitting wallets capable of providing such third-party attestations to deposit into markets.

3.3 Interest In Different Tokens

At present, each market 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 lending-related protocols), but may give rise to issues for the borrower in the situation where they cannot acquire enough of a niche 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 reliable, secure feed for said native token in order to establish the correct amounts to release. Interesting, but dangerous.

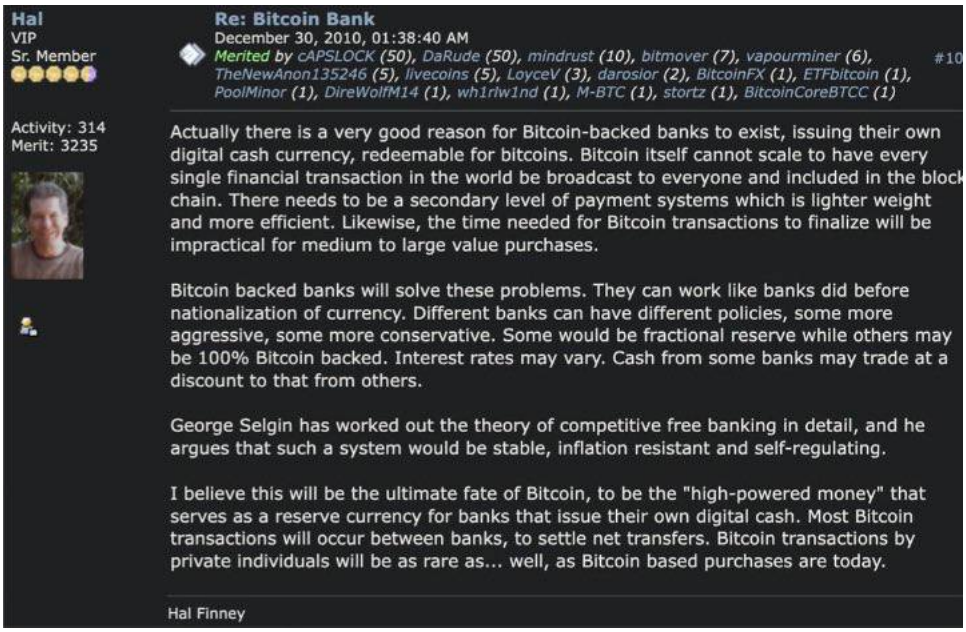
4 Code Repository

The code for the Wildcat protocol is available under a source-available Apache 2.0 with Commons Clause license, and can be accessed - along with the most recent version of this whitepaper and an SDK - at:

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

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)

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 1.0 [13 November 2023]
  - Clarified market token mechanics in withdrawal mechanism
  - Emphasised protocol fees are not tied to market tokens
  - Presented new sentinel mechanism (previously an empowered EOA)
  - Detailed powers of archcontroller owner
  - Flagged lack of support for rebasing ERC20 underlyings
  - Substantial rephrasing and corrections throughout
- 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