The Wildcat Protocol V2: Wildcat But Worse

Wildcat Labs

October 21, 2024

Abstract

In this paper we present Wildcat V2, a protocol that permits borrowers to establish highly customisable fixed-rate, undercollateralised credit facilities on Ethereum. Wildcat V2 markets are adjustable in asset, capacity, interest and penalty rates, reserve ratios, withdrawal cycle length, minimum deposit amounts, token transferability, lockup duration and more.

The primary improvement upon Wildcat V1 is the introduction of pre-transaction hooks, which can be used to both extend functionality and gate access to core functions per borrower requirements. Zero-knowledge proofs of off-chain data, soulbound token holdings, membership within a Merkle tree of a list of known counterparties: all of these plus others are now supported as mechanisms to grant deposit credentials.

Credit can now be extended and legally underpinned without the counterparties ever having made contact, provided that the lender finds the terms (and borrower) acceptable.

Wildcat V2 is aimed at organisations such as market-makers and trading desks wishing to borrow digital assets in the medium-term, as well as DAOs and DeFi protocols wishing to raise funds without first having to sell a native token into available liquidity. Auxiliary contracts that liquidate arbitrary assets to cover delinquent debt can be bolted onto markets.

Wildcat is fundamentally permissioned in nature: the protocol operators dictate who can act as a borrower, and in turn borrowers dictate the terms of the markets they deploy and any access constraints. Given that the positions that Wildcat markets allow a lender to enter are undercollateralised by design, discretion should be exercised by lenders. Defaults are a distinct possibility, and Wildcat does not offer protection against this risk.

Here we describe - in brief - our reasoning for creating Wildcat, a high-level technical sketch of the V2 implementation, a description of the lifecycle of a market, the various scenarios users can encounter, and some ideas of where we are considering taking the protocol next.

Contents

1 Introduction

1.1 Motivation

This is a whitepaper about on-chain undercollateralised lending. DeFi and crypto writ large doesn't feel like it 'suits' this field due to the pseudonymous nature of it all - we laud Aave, Euler, Morpho et al for their solutions to over-collateralised 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.

Credit is notionally a fairly boring topic compared to things like asset fractionalisation or shoehorning an LLM into an NFT somehow, but credit facilities are also one of the most important growth drivers of an ecosystem. 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 Clearpool, Goldfinch, TrueFi, Centrifuge, Atlendis and even Aave with its credit delegation (Maple/Syrup has migrated to secured lending since the release of Wildcat V1). These protocols serve their users well, but we believe that they have gaps or constraints in their product offerings that prevent them from scaling into truly powerful credit mechanisms. We believe Wildcat addresses these in a way that makes it a more appealing choice for borrowers and lenders alike.

The demand for credit is elastic in the face of real interest rates, and the circumstances surrounding that demand even more so. Protocols that only support a small set of borrowable assets and/or dictate the rate, tenor and repayment schedule under which an entity can borrow feels like leaving some of the flexibility of DeFi on the table, in the name of providing credit in a way that won't frighten a TradFi broker. More promising – to our eyes at least – is to leave things to the borrower to ask and candidate lenders to bid if the request passes their smell test. That bar needs to be higher post-FTX, but there are several mechanisms now to provide information on liquid reserves and credit risk reports for would-be borrowers.

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, their inability to provide a bespoke mechanism often drives would-be users into off-chain deals, where we have to wait 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.

We think DeFi and its participants are both mature enough to practice self-awareness when it comes to credit, and we built Wildcat in response.

1.2 Features of Wildcat

The pieces that are novel to Wildcat 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 nearly any non-rebasing ERC-20 token,
- Borrowers can determine the reserve ratio of a market on creation, which impacts its required reserves (the assets within a market that cannot be borrowed). In homage to the American central banking system, Wildcat V2 permits the reserve ratio to be zero.
- The maximum capacity of a market – the amount of debt it can take on – is initially dictated by the borrower and can be adjusted at will,
- Fixed interest 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 engage (and downwards, subject to reserve ratio requirements discussed later),
- 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 insufficient required reserves incur a penalty APR that indicate the strength of a borrower's intention to maintain promised levels of collateral, and
- Borrowers can close a market at any time (subject to configuration), 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.

The above relate to the internal parameters of the market, and are consistent with the features of Wildcat V1. More generally, Wildcat V2 introduces hooks which permits arbitrary checks on certain core functions, such as:

- Deposits are constrained to those addresses that have gained deposit credentials, which may have expiries attached to them. These are gated by *role provider* contracts which can be added or removed by the borrower, and each provider dictates a different methodology for gaining access. Wildcat provides a library of role providers that can be bolted in, but the check imposed is arbitrary.
- Deposits can also be rejected if they are not above a certain limit, reducing the administrative hassle that arises when someone deposits assets which are not 'worth' borrowing but still accrue interest (i.e. dropping 1,000 USDC into a market for a fund).

- Withdrawal requests can be gated until a market has existed for a certain amount of time, enabling a simulacra of fixed duration markets. For example, a borrower could gate withdrawals for a year after deployment, after which the hook-constraint ends and the market converts to open term. Various switches are available to the borrower on deployment to disable the ability to prematurely terminate markets or reduce the term.

- Transfers of the debt token issued by a Wildcat market can be constrained: they can either be fully transferable, transferable only to addresses that either a valid deposit credential or are 'known lenders', or completely restricted between the market and the address that deposited the corresponding assets.

From the perspective of lenders, once they have obtained a credential from a role provider (and for as long as it remains valid, although they can be renewed), their experience will look much the same as it currently does in a DeFi vault. There are also lender-oriented defences built-in via both a penalty rate for borrower delinquency and constraints on the borrower when reducing APR:

- Lenders can deposit to - and withdraw from - any market for which they have an valid deposit credential, and a credential from one role provider may grant access to multiple markets simultaneously, depending on borrower configuration.

- Lenders that previously held a deposit credential and either deposited into a market or received market tokens while it was valid will be able to make a withdrawal request even once it has expired. Withdrawals cannot be requested while a market is in a fixed term state, and they always can if a market has been closed.

- Lenders benefit from a penalty APR that triggers after a market remains delinquent (holds less than its required reserves) for a rolling length of time beyond a borrower-configured grace period. This penalty – while active – is distributed to all lenders, and no part of it is receivable by the Wildcat protocol itself.

1.3 So, Who's This For?

In a sentence: 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, Wildcat 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 when called upon so you can withdraw your notional plus interest.

With that said, who do we picture is sitting up and taking notice of this? We envisage a handful of cases:

- Market makers looking to provide order-book liquidity, who can make more profit off the spread than they are offering in interest.
- Funds and trading desks looking to scale profitable strategies.
- DAOs/protocols looking to raise funds for development/operational purposes: most of these have their treasury value denominated in a native token and cannot (or will not) sell said token for assets that are more commonly accepted due to the potential price impact,
- Entities with non-productive assets that they that seek to utilise in a yield-bearing manner by providing credit facilities to borrowers,
- Off-chain entities looking 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 freely-transferable market tokens issued by a borrower which can 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, although it can reasonably be described as falling into the RWA bucket by virtue of facilitating private credit: 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.

Here's how this actually works under the hood.

2 Protocol Details

2.1 High-Level Overview

There are five main components to Wildcat V2: the *archcontroller*, *hook factories*, *hook templates*, *role providers* and the *markets* themselves:

- The archcontroller acts as a global registry of deployed contracts and provides access control for borrowers,

- Hook factories maintain a list of approved hook templates, and acts as a launchpad for markets and the hook instances that they depend on (for those familiar with Wildcat V1, they are roughly equivalent to market controllers).
- Hook templates are smart contracts which gate various core functions of markets such as depositing, withdrawing and adjusting APRs. Templates are cloned into hook *instances* as required to attach themselves to markets upon deployment of the latter. A template must be registered with the hook factory by the archcontroller owner before it can be used.
- Role providers are accounts which can either be queried by a market (pull-providers, typically contracts) or explicitly tell a market (push-providers, such as the borrower EOA itself) about whether or not an address should be given a credential which enables them to deposit underlying assets.
- Markets are the user-facing 'result' of the protocol - smart contracts that mint and burn rebasing ERC-20 tokens representing claims on debt.

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 markets through hook factories. Removed borrowers can continue to interact with any of their existing markets.
- Registering and removing hook factories. Removal of a hook factory prevents any further markets from being deployed from it, but leaves existing ones untouched.
- Registering and removing hook templates. Removal of a hook template prevents any further hook instances from being cloned from it, but leaves existing ones untouched.
- Removing markets, which does not hinder their operation in any way, but rather hides them from an SDK that reads from the archcontroller for the purposes of displaying on any front-end which makes use of said SDK.
- Updating the protocol fee configuration for existing and future markets in hook templates (maximum value of 10% of base APR).
- Update the address of the SphereX engine (on-chain exploit prevention) associated with the protocol deployment.

The only meaningful powers the archcontroller owner has from the perspective of active markets is deciding who can be a borrower in the first place and setting the current protocol fee. It can cease future expansion of their Wildcat protocol deployment by removing all hook factories, but cannot interfere with already deployed markets.

A protocol kill-switch does notionally exist: updating the SphereX engine to a contract which rejects all incoming calldata. We will mitigate this power shortly after deployment on mainnet by locking the ability to swap the engine while still being able to whitelist novel calldata patterns to account for new hook templates and market structures.

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.* At no point can the protocol operators access (or take custody of) any assets within a market beyond the fees incurred by the borrower.

This means that any credit extended/received within a market *cannot be modified by third parties.* There is one exception: later in this paper we discuss the *sentinel* – an immutable contract defined by the market itself that the protocol operators cannot update –, which has the power to shift market-associated balances of a lender to an auxiliary escrow contract in the presence of sanctions.

2.1.2 Hook Factories

Hook factories are the staging ground of Wildcat V2, replacing and significantly extending the market controllers of V1. They contain the initcode and hashes for the template contracts of the markets they ultimately facilitate, as well as storing the addresses of hook templates currently registered with them. When a borrower deploys a market they do so through a hook factory, passing parameters through in a way that identifies which templates need to be cloned into instances and how they must be configured.

The core functionality of a market is defined by the aforementioned initcode, while the bells and whistles such as fixed term durations, transfer restrictions, deposit credential eligibility and minimum deposit requirements are now all defined by hook instances. While Wildcat V2 is launching with a single hook factory and a selection of templates, this design means that new functionality (and indeed new market structures) can easily be facilitated under a single archcontroller, allowing a hydra-like deployment of a Wildcat protocol instance.

2.1.3 Hook Templates/Instances

Hook templates are the base contracts that dictate how access to core functions within a Wildcat market can be accessed. They define what yes/no questions need to be answered in order to 'punch through' to the market code itself. As an example, the *AccessControlHooks* template available at V2 launch dictates how to add and remove role providers, how role providers are expected to respond to queries about market eligibility for a would-be lender, and how long deposit credentials last until they expire. In turn, the *FixedTermLoanHooks* template permits a borrower to prevent withdrawal requests for a configurable period of time, as well as holding a number of flags a borrower can configure determining whether they are allowed to prematurely convert a fixed term market into an open term one or terminate a market that is in fixed term.

When a new market is created, a borrower has the option of either cloning the hook templates necessary for their desired configuration into instances which they own and are bound to said market, or tying the market to existing hook instances in order to route everything through a single point of control. For example, a borrower may wish to make use of a single well-configured setup for six different markets covering separate assets.

In order to not make the protocol UI overly complex, we have made an assumption which defaults to this latter case: a borrower that deploys multiple markets has their markets tied to any existing hook instances that they have deployed.

2.1.4 Role Providers

In an access control hook instance, a borrower configures a set of role providers - accounts which grant credentials to would-be lenders. Each role provider defines whether it is a *pull provider*, meaning whether the hooks contract can query the role provider to check if a lender has a credential, or a *push provider* which directly pushes credentials into the hook instance.

Each role provider is configured by the borrower with a TTL (time-to-live): the amount of time a credential granted by the provider is valid, subsequent to which the credential must be renewed, either through the same provider or through any alternative provider that the borrower has chained to the access control hook instance. It is worth noting that credential expiry only applies to deposits: assuming the lender deposited or received tokens while the credential was valid, they will *always* be able to make a withdrawal request, unless they are subject to sanctions or the market is currently within a fixed term.

These role providers can either be constructed, deployed and registered by the borrower themselves, or they can opt to clone a role provider template (which works in a similar fashion to how hook templates/instances work, so we won't relitigate it here). The interface requirements are sufficiently light as to support arbitrary qualifying mechanisms depending on borrower needs: simply checking for OFAC compliance, proof of being within a certain Merkle tree, holders of a soulbound token, all the way through to requiring credentials from an on-chain KYC provider or a ZK proof of being able to log in to a specific website.

2.1.5 Markets

The markets are the important part 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. We'll use every single bell and whistle here to illustrate how configurable a Wildcat market can be.

A market is deployed from the hook factory with the following parameters (the UI makes this a relatively simple experience of checkboxes and dropdowns):

- Underlying Asset: `WETH`
- Symbol Prefix: `whc`
- Name Prefix: `West Ham Capital`
- Maximum Capacity: 1,200 `whcWETH`
- Reserve Ratio: 10% of `whcWETH` supply
- Base APR: 5%
- Penalty APR: 10%
- Grace Period: 5 days
- Withdrawal Cycle: 12 hours
- Protocol Fee: 10% of Base APR (0.5%)
- Minimum Deposit Amount: 50 `WETH`
- Market Type: Fixed Term
- Fixed Term Maturity: 6 months
- Transfers Permitted: No
- Force Buyback Enabled: Yes
- Can Reduce Fixed Term Early: No
- Can Terminate Market Early: No
- Role Providers: Open Access (TTL: 604800)

The result is a newly created market that accepts `WETH` and issues a market token with symbol `whcWETH` in exchange (the market token name is *West Ham Capital Wrapped Ether*).

In terms of lender access - entities interested in extending credit can request a credential from the open access pull provider, and presuming that their address is not on the Chainalysis sentinel contract they will receive a valid credential that expires in one week (whereupon they will have to request a new one). This is a silly example given that the credential will always be auto-renewed unless the address is sanctioned, but one can imagine a role provider that requests occasional re-upping of on-chain KYC credentials with, e.g. a three-month TTL.

Any deposits made by lenders to this market cannot be requested back until six months have passed *after market deployment* - someone who turns up after the market has existed for five months and makes a deposit (assuming there is sufficient capacity) will be able to make their withdrawal request/s one month later once the market reverts to open term. Once a market reverts to open term, it cannot be converted into a fixed term again. This market was initialised by the borrower to not permit the borrower to reduce the maturity while it is within the fixed term configuration. Note that the maturity of a fixed term loan can only be reduced, never extended. Conversely, the APR of a fixed term loan can only be increased in a fixed term: decreases are only possible once a market has converted back into an open term.

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

- If the market had a maximum capacity of 1,000, the borrower could only withdraw 900 `WETH` maximum, since 10% of the debt must remain within the market as liquid reserves,
- If the borrower configured the maximum capacity to 1,111 to withdraw up to 999.9 `WETH`, the market would find itself delinquent nearly immediately given the accrual of protocol fees,
- By contrast, a maximum capacity of 1,200 allows for 1,080 `WETH` to be withdrawn if the market is at full capacity. If the borrower constrains themselves to borrowing maximum of 1,000 `WETH`, there is at least *some* buffer space beyond the required reserves for interest accrual and for lenders to place withdrawal requests against.
- Of course, the borrower could always opt to select a 0% reserve ratio instead, and not have to worry about this beyond keeping sufficient funds in reserve to account for protocol fees that accrues over time. The caveat here is that that once the market converts to open term, they will need to either monitor the market closely in order to account for incoming withdrawal requests, or terminate the market outright (more on this later).

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' the borrower is 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. Enabling force buybacks enables a pressure-reduction valve for outstanding debt, but may lead to off-chain pressure from large lenders in situations where the borrower is prevented from triggering one. Eliminating the ability to reduce a fixed term maturity is appealing to lenders, but may lock the borrower into paying interest on more than they can handle if they misjudged their maximum capacity and received what they asked for. And on and on it goes.

In the 'boring' case, 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 a 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 detail here: you can read more on the protocol Gitbook: there is a link to this at the end of the paper.

Another specific figure worth highlighting here is that of the protocol fee, which is defined here as 10% of the 5% base APR. This 'additional' 0.5% APR interest is claimable by a separate *fee recipient address* defined by the protocol. Note that the presence of a protocol fee is not guaranteed, and depends on the protocol fee configuration of the hook factory from which the market is derived. Protocol fees do not factor into the rebasing rate of market tokens: they are tracked separately as underlying assets and claimed from available 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 this market only receives 10 `WETH` in deposits, the borrower only needs to maintain slightly over 1 `WETH` 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 Edits Role Providers

As the owner of the access hook contract associated with one or more markets, a borrower has free reign to add and remove role providers, each of which provides a unique path to granting deposit credentials to would-be lenders.

This does not matter to *existing* lenders, as once an account has deposited underlying assets to a market while holding a valid credential, they will be marked internally as a 'known lender' and granted the permanent ability to place withdrawal requests (while non-sanctioned, once fixed term has passed).

The known lender status is also granted to accounts that receive market tokens from elsewhere while holding a credential, although this is not possible in our example market as it has disabled transfers: meaning that market tokens can only flow to-and-from the market and an initial depositor. This means that wider DeFi integration is not possible for this market (no LPing the debt, for example), but this is a choice left to the borrower. The other two options are 'no transfer restrictions' and 'restricted to credential holders/known lenders'.

If the borrower in this example elects to remove the only role provider being used (open access), then once all extant credentials have expired, no further deposits can be accepted. However, lenders that are currently within the market are able to leave once the six-month fixed term maturity has elapsed. Similarly, if a new role provider is added, this doesn't impact anyone until new lenders start obtaining credentials through it.

This is a somewhat trivial example given that an open access provider is a superset of everything else, but picture a whitelist provider being added alongside one that grants access to Binance Account-Bound Tokens and you get the idea.

2.2.2 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 at will. This reduction can be to a value below the current outstanding market token supply, but this does not eliminate any outstanding debt above the new level - rather it indicates that no new deposits will be accepted until the market token supply is below this level as a result of either withdrawal requests or force buybacks.

2.2.3 Borrower Adjusts Base APR

A borrower that decides to increase the base APR to encourage would-be lenders is able to do so at will, up to the maximum permitted by the hook configuration of the market. If the borrower of the aforementioned market decides to increase the base APR from 5% to 9%, this is fine. If there is a protocol fee in place for the market, the final amount owed by the borrower will take the new APR into account: with a protocol fee of 10% of base APR in place, making this change will bump the 'true' APR that the borrower pays from 5.5% to 9.9%.

A borrower that wishes to *decrease* the base APR is capable of doing so, subject to several constraints. Firstly, the APR cannot be reduced in a fixed term market. Secondly, the APR cannot be reduced while a market is delinquent. And thirdly, assuming the market is healthy and open-term, a borrower may have to return assets to the market in order to meet a temporarily increased higher reserve ratio when making the reduction.

Allowing a borrower to arbitrarily reduce the base APR 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 drops the APR to near-zero immediately afterwards to effectively eliminate future interest.

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 30% 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:

- If the base APR of a market is reduced by 25% or less in a two week period, the reserve ratio remains unchanged.
- If the increase is in excess of 25%, the reserve ratio is set as the larger of the *existing* ratio or double the relative rate change (capped at 100%): $NewReserve = max(max(100\%, 2 * \frac{OldRate \ - \ NewRate}{OldRate}),\ OldReserve)$.

This means that reducing the base APR from 5% to 3% in a market with a 10% reserve ratio effects a new ratio of 80% regardless of the underlying asset. However, reducing it to 3.75% would effect no change, and two weeks later the APR can be reduced further.

This also means that a borrower that reduces the base APR by 50% or more in one shift must fully collateralise their market. After the temporary limit has expired, any address can *reset* the ratio back to its previous value (deleting an internal struct that puts the higher ratio in place) provided that the market is not delinquent under the higher ratio when doing so. The goal here is to provide lenders as much flexibility to exert their ragequit option as possible.

In the event that a borrower regrets their reduction, they are capable of erasing the increased reserve ratio by changing the base APR back to within 25% of the initial value. If the rate is changed to a value between the two, then the increased reserve ratio remains in force but is reduced in magnitude. To use our example above, if the borrower increased the APR to 3.75% the reserve ratio would reset to 10% immediately, but if they changed it to 3.5%, the reserve ratio would drop from 80% to 60% without impacting the two week timer.

A borrower reducing the APR by 25% in one transaction and then immediately attempting to reduce it by another 25% will find themselves disappointed: Wildcat markets track these, and measures all changes against the initial base APR until two weeks has passed from the first shift (unless it is being increased relative to the current value).

2.2.4 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. Six months later once the fixed term elapses, 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 due on Bob's deposit from the moment that his `whcWETH` tokens are 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 15.5% (5.5% base plus protocol fee, 10% penalty).

Note that the protocol fee does not take the penalty APR into account, as Wildcat generating additional revenue from markets that are in penalised delinquency would give rise to all manner of perverse incentives.

It is worth noting that market state needs to be updated in order for delinquent or penalty states to 'register' within the contract: this is done through a deposit, a withdrawal request, a claim execution or simply by invoking a heartbeat function called `updateState`, which could easily be handled by a sentinel. Once such an update takes place, any additional interest due is calculated retroactively by tracking timestamps and appropriately updating the scale factor.

2.2.5 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 six months later, 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 and 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 200 `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.

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.

2.2.6 Borrower Removes Access From An Active Lender

As we have seen, lenders in Wildcat V2 markets obtain deposit credentials from role providers, and can subsequently deposit to any markets that the borrower deploys using the relevant access hook instance. The borrower has the ability to both revoke a specific credential (e.g. to fully phase out credentials issued by a role provider that has since been deregistered) and – more seriously – to block an account from obtaining a future credential from any role provider. Such blocks are reversible, should the borrower choose.

In the event that a borrower elects to erase a credential or block a lender, they are - unsurprisingly - barred from depositing further to any of the markets tied to that access hook. However, a known lender to a market can make withdrawal requests at their leisure, short of being subject to a forced buyback.

2.2.7 Borrower Wishes To Reduce Their Outstanding Debt

One of the more nuanced features added to Wildcat V2 beyond the introduction of hooks is the ability for borrowers to effect a force buyback of market debt. This power is enabled by a flag that must be selected by a borrower on market creation, and permits them to reduce their outstanding credit line by directly exchanging market tokens held by known lenders for an equivalent amount of the underlying asset held by the borrower.

In effect, this is a speed-run of the lender making a withdrawal request and subsequently making a claim, with the difference being that the underlying assets do not touch the market itself, and the market tokens are not burned. Triggering a force buyback for the first time in a market immediately marks the borrower itself as a known lender, and notifies lenders via the protocol UI that a force buyback has occurred. In order to prevent sudden, violent jolts to the required reserves of a market, borrowers executing a force buyback must subsequently make a withdrawal request themselves to burn the market tokens and reduce the outstanding supply.

In order to prevent force buybacks from being used as a preferential repayment mechanism for lenders if a borrower sees an iceberg up ahead, they cannot be used a) while a market is delinquent or b) if a market was initialised as fixed term (even after it has converted to open term).

If lenders get the jitters and start requesting withdrawals, force buybacks are unavailable as a queue jump. Similarly, a borrower that deployed a fixed term market (offering a non-decreasing rate for deposits for a fixed time) may only access force buybacks by closing the market first, assuming they have not configured their market to prevent early termination.

2.2.8 Borrower Wishes To Close The Market

Subject on configuration, a borrower has the right to close their market: indeed, we expect that all created markets will eventually be closed unless borrower keys are lost. This is an edge case of the borrower electing to reduce the base APR - the 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 market parameters can be made. It is then up to the lenders to make withdrawal claims for their assets, or alternatively, a borrower can effect force buybacks for every lender – if enabled – and close out all the outstanding debt themselves.

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.

2.2.9 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 (or borrower, depending how it's configured) could replace the borrower of a fully-subscribed market with 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 via the `repay` function. 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 obligations from another address and all lenders placing full withdrawal requests.

2.2.10 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.

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

2.2.11 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 their address is added to the Chainalysis sanctions oracle.

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

In the event that a 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 force them into a withdrawal request and create an escrow contract to subsequently hold the underlying assets that would be transferred to them on claim.

Within each market contract, there is also a 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 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.

2.2.12 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 (this is fundamentally an off-chain issue), and would advise speaking to lawyers if this happens.

2.2.13 Borrower Simply Doesn't Repay Their Debts

The operators of Wildcat do not underwrite any markets deployed via the protocol, nor do we attest to their risk levels in any way, shape or form. We seek to treat counterparties as rational entities: if a borrower is offering conditions too good to be true but is a complete unknown, or is unable or unwilling to provide evidence of their financials, reason dictates that you should not engage with their markets. You extend credit via Wildcat at your own risk.

Credit is fundamentally a risky premise, and on-chain credit is no different: defaults are to be expected in aggregate across an ecosystem. Wildcat – the protocol or its operators – will not make you whole given a default, as defined by a loan agreement that you may be offered by a borrower via the protocol UI.

It seems paranoid to belabour the point, but we do need to be very clear: your capital is not shielded by Wildcat in the same way that it is in a protocol such as Aave or Euler, and your assets are returned by the grace of your borrower, which may require compulsion by a court in the worst instance. Please be careful using it, and treat it with the seriousness that it deserves.

To answer the actual question, however: if the market configuration permits token transferability, lenders can sell (or otherwise exchange) their market tokens to entities that are willing to follow up in court for recovery if they aren't willing to do so themselves.

And that's all she wrote.

That's how you utilise Wildcat V2. Any other market parameter we haven't mentioned in an adjustable scenario (beyond APR and capacity) cannot be changed while a market is live, and the operator of the protocol deployment itself does not custody or have control of your funds at any point.

We hope that what we have built is useful to you.

3 Future Directions

Despite this being the paper that announces Wildcat V2, there are other things that we want to see in the protocol but were prevented from adding, either because of time, complexity or the contract code-size limit introduced by EIP-170.

Here's a non-exhaustive list of things we're thinking about:

3.1 Auxiliary Market Collateral Contracts

Wildcat currently requires borrowers to make timely repayments within their grace period to markets whenever they fall into delinquency so that they don't fall into the penalised state, which demands a certain amount of maintenance. Moreover, given that Wildcat does not demand collateral from a borrower upfront, it can be hard for a lender to determine what a borrower actually has in assets to illustrate their solvency, or what they are 'borrowing against'. We leave the choice of how much information a borrower wishes to broadcast about their financial situation to them (either through proofs of reserves, credit risk reports etc), but there is a more direct way to go about this.

It is possible to attach a contract to each market (either on deployment or retroactively) which contains arbitrary collateral, which can only be reclaimed upon the closure of a market. Provided that there is a path – perhaps via an aggregator – to convert that collateral into the underlying asset of the market, such a contract can be configured to be partially liquidated and transferred to the market as a repayment every time the market hits penalised delinquency, on a cooldown timer the length of the grace period.

Such a mechanism would permit DAOs to put up native tokens against their credit, without actually selling them (if no path exists, or the slippage would be too high, the liquidation wouldn't trigger). The effectiveness of this would greatly vary of course, depending on the on-chain liquidity of the collateral (c.f. a thinly traded governance token versus wrapped Ether), but there is promise for native tokens of non-Ethereum chains such as NEAR and DOT by making use of protocols such as Universal.

In any event, it would be extremely useful to allow a borrower to lock up cbBTC, AAVE or what have you in order to illustrate their capacity to repay a credit line in stables. Given that the implementation of this is not contingent on modifying the Wildcat protocol itself, this is something of a priority.

3.2 Automated Ragequit On Reduced APR

One feature *potentially* enabled by the introduction of the force buyback functionality in Wildcat V2 is that of allowing lenders to specify the minimum interest rate at which they are willing to lend to a market. Instead of relying on lender vigilance (or notifications from a bot monitoring a market) to spur their withdrawal request if they decide the terms are no longer acceptable, this feature could permit a variant of a market whereby a borrower that seeks to reduce the base APR in an open term market must simultaneously execute a force buyback of all debt belonging to lenders for whom the rate is unacceptable.

This may well be possibly by introducing a hook template with a modified hook for force repayments, but would require a per-lender parameter which contained the rate at which lenders want out.

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.

3.4 More Traditional Term Structures

At present, Wildcat markets can be in one of two modes based on the hook templates and market configurations available: fixed term or open term. The former prevents any withdrawal requests and does not ask for any lender repayment until a given amount of time has passed, and the latter is a free-for-all whereupon a lender can request their assets back a block after depositing them.

This works for most purposes, but we can envisage more traditional borrowers and lenders alike that would prefer a fixed term mechanism whereby a set amount of interest periodically comes due (say, once a month), with the balance due at maturity. We do not support this at present, although it might be possible by adjusting hooks attached to borrow and deposit functions that prevent interaction unless a repayment has been made within a set timeframe.

With that said, the nature of Wildcat markets is that they accept rolling deposits up to capacity (a borrower is not issued a lump sum up front unlike in other credit-oriented DeFi protocols), so calculating how much needs to be repaid in a given period may be something of a moving target that is messy to calculate via independent on-chain code. Nonetheless, we want to think about it in time.

4 Repositories

The code for Wildcat V2 is available under a source-available Apache 2.0 with Commons Clause license, and can be accessed at:

`http://www.github.com/wildcat-finance/v2-protocol`

The most recent version of this whitepaper can be accessed at:

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

The Wildcat SDK can be accessed at:

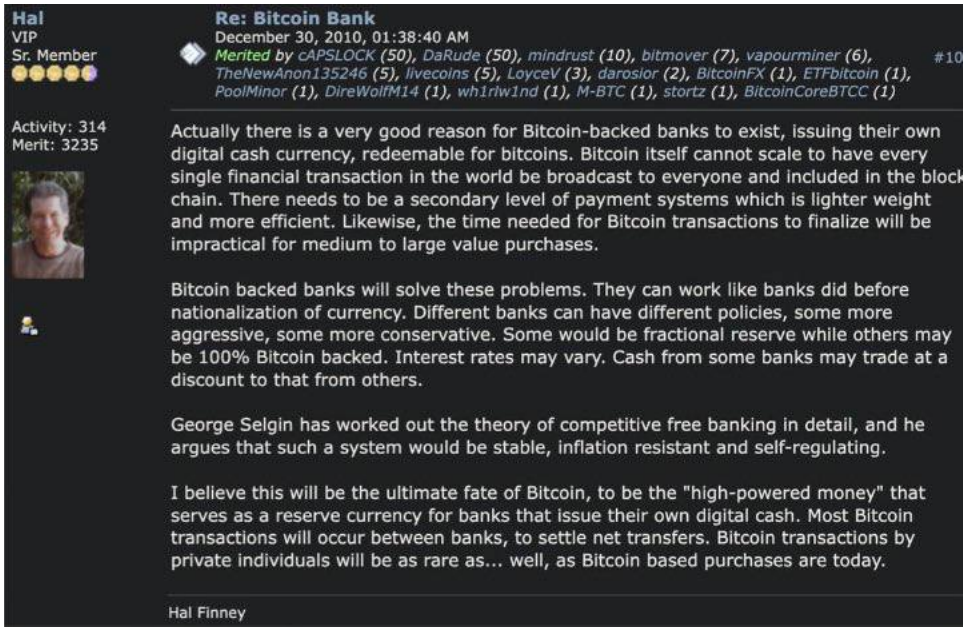
`https://github.com/wildcat-finance/wildcat.ts`

Broader protocol documentation, details of security reviews, risk disclosure statements plus auxiliary agreements and templates can be accessed at:

`http://docs.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)
#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

- Version 2.0 [21 October 2024]
  - Introduced pre-transaction hook mechanism
  - Introduced hook templates (e.g. access control, fixed terms)
  - Introduced role providers for deposit credentials
  - Introduced forced buyback mechanism
  - Replaced market controller factories with hook factories
  - Eliminated ability to reduce APR while delinquent
  - Polished base market structure
  - Substantial rephrasing and modifications throughout
- 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