The Wildcat Protocol: Banking, But Worse

Laurence E. Day, Evgeny Gaevoy & Dillon Kellar

Whitepaper v0.1

May 15, 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ratified borrowers to establish fixed-rate, on-chain credit facilities, the collateral of which can be partially withdrawn for the borrower's purposes.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, far more general, and designed by maniacs.

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

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

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

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

No, there isn't a token right now.

Please stop asking for an airdrop.

Contents

1 Introduction

1.1 Motivation

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

Sometimes, we need to trust.

There are several protocols in operation at the time of writing that enable users to lend their funds in an undercollateralised manner to third parties who utilise them either on- or off-chain, such as Maple, Clearpool, Goldfinch, 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 tend to offer 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 highly liquid and easy to ramp on and off exchanges. This works well enough, 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 the release of the Arbitrum 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` (incurring slippage both ways if they don't strike an OTC deal) and then hope that their fees taken from market-making `X` exceeds the sum total of said slippage, 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 yield (APY) on these deposits - whatever their underlying asset - is typically decided by way of a utilisation-based rate curve, whereby the interest paid by the borrower increases as the withdrawn funds approach a vault's capacity: the maximum amount that they can borrow. The exponential rate that these curves often adopt towards the final quartile can lead to perverse incentives whereby borrowers seek to take out a loan of more than they require and rehypothecate the excess to 'reduce' repayable interest.

Notwithstanding the moral hazards here where borrowers often 'second-hop' this excess capital into other lending protocols - often farming protocol token incentives as a result -, utilisation-based APYs mean that there is little certainty up front for a borrower in terms of what to repay. Similarly, a lender who contributes their capital to a popular vault at a certain rate can easily find themselves surprised if the borrower repays most of the outstanding balance shortly thereafter, dropping a utilisation APY to a point where they could better utilise those funds elsewhere. Depositing assets into a vault with a limited set of borrowers doesn't give rise to a mechanism to correct capital availability based on demand - your assets are deposited there exclusively for them, and their usage isn't directly parameterised by market sentiment or activity. In our opinion, in situations such as this you should be getting paid a fixed rate whether they're being used or not.

Finally, existing loan-facilitation protocols often use tools such as Chainalysis or Credora to analyse the addresses of participants for proximity to sanctions (this is good) and a measure of credit-worthiness (this is less good). However, fine-grained, proactive control of approved lenders and other such vault parameters from the perspective of a borrower is often scarce. In particular, the maximum amount borrowers are allowed to take up (as well as the terms) is currently often determined by an in-housed 'pool delegate' or third-party protocol, impinging on the freedom to transact in a situation where a sufficiently grave legal agreement is perfectly capable of constraining carefully selected borrowers to working within the bounds of sanity. The involvement of latter parties such as these also give rise to a requirement of vault deployment and management fees to remunerate their labour, fees that we believe could be better used by a sophisticated borrower by in-housing these checks as part of their own processes.

1.2 Features of Wildcat

The pieces that are novel here fundamentally relate to the freedom of the borrower to dictate the terms of the vaults that they deploy:

- Vaults can be created for *any* ERC20 that the borrower wishes to acquire,

- Borrowers can determine the liquidity ratio of a vault on creation (i.e. the maximum percentage of deposits they can withdraw, although interest is charged on the full balance of vault deposits),
- The maximum capacity of a vault can be adjusted at will up or down by the borrower depending on their current need (subject to available liquidity ratio requirements, which we describe presently),
- Yield rates are chosen at deployment and fixed, but can be adjusted upwards by the borrower in order to incentivise lenders to participate (and downwards, again subject to collateral requirements),
- Borrowers determine the length of the withdrawal cycle: the rolling period during which multiple lenders will have their redemption amounts prorated if their sum exceeds the available vault collateral,
- Borrowers can choose a grace period after which vaults that have an insufficient liquidity ratio incur a penalty APY (selected at vault deployment) 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 a given vault, subject to their own KYC processes, and
- Borrowers can terminate a vault at any time, dropping the APY to 0% while simultaneously returning all outstanding collateral and interest: this blocks their ability to withdraw, and allows lenders to exit at their leisure.

From the perspective of lenders, their experience will look much the same as it currently does, with protections built-in via both a penalty rate for borrower delinquency and a legal agreement that all participants attest to via gasless signatures before being permitted to engage:

- Lenders can deposit to - and withdraw from - any active vault deployed by a borrower that has approved them via a 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 APY that triggers after a vault remains delinquent (under its liquidity ratio) for a rolling length of time beyond the aforementioned grace period. This penalty is distributed to all lenders, and no part of it is receivable by the Wildcat protocol itself.

More generally:

Vaults do not have a minimum period must be locked up prior to redemption: lenders can queue up a withdrawal of their deposit - plus interest - at any time, with amounts received subject to potential proration given circumstance.

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

As alluded to above, vaults are governed by a legal agreement between each borrower and their lenders on a per-vault basis, requiring unequivocal agreement to terms such as the jurisdiction in which to bring arbitration proceedings and/or civil claims against a borrower, and the conditions that terminate obligations.

1.3 So, Who's This For?

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

"People are just going to set up vaults, take the deposits and run", we see you tweeting. Sure, perhaps. That's been happening since money existed, and that's why you introduce the legal system as a backstop. Our vision of decentralised finance has always involved the cutting out of rent-seeking middlemen and increasing capital efficiency, not a wholesale rejection of the non-financial systems that bind us all. We see no paradox in allowing credit facilities to be brought on-chain with the power of a gavel behind them, and we would much rather allow the ability for deals such as these to be brought into the light - with terms that all can see - than not.

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 denominated 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 (that’s probably you reading this) that have nonproductive assets without native yield-bearing variants that they are willing to utilise in the form of credit to a borrower provided they are willing and able to pass their KYC checks,
- Off-chain entities as borrowers that wish to raise funds on-chain that do not - or cannot - interface with the traditional banking system or institutional crypto platforms, and
- Third-party protocols that may be interested in listing vault tokens issued in the name of a borrower whereby they trade at a discount or premium reflecting market confidence in the borrowers ability to repay their loan.

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

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

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

2 Protocol Details

2.1 High-Level Overview

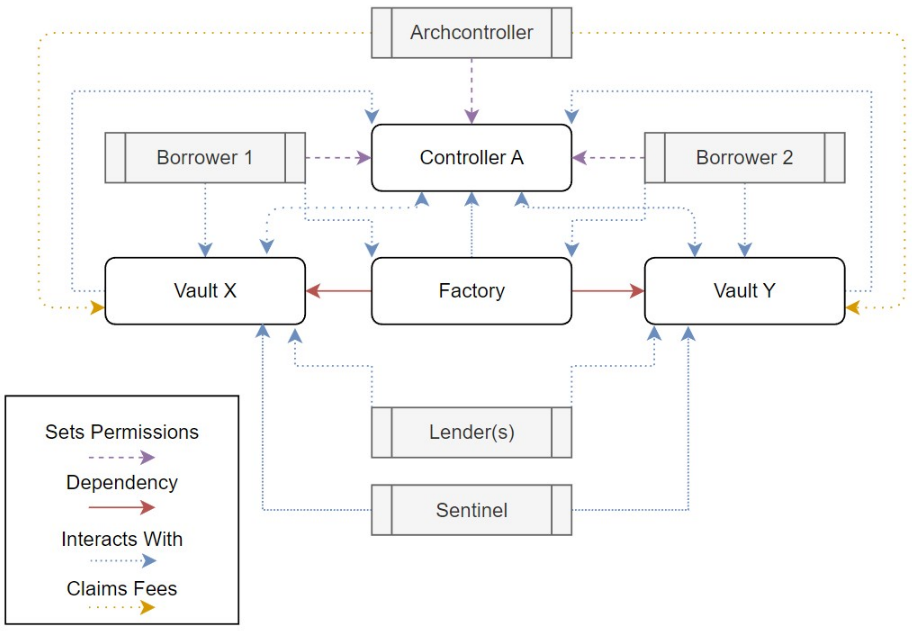
There are three main components to the Wildcat protocol: the *controllers*, the *factory*, and the *vault tokens* (or rather, the *vaults*):

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

The address that owns a given controller is referred to as an *archcontroller*. This address represents the entity responsible for dictating the borrowers that can deploy vaults using said controller, and will typically be operated by the Wildcat protocol itself. Another Wildcat-operated multisig exists - the *sentinel* - that has restricted powers to wipe out the outstanding balance of a lender for a vault (which we will address, before you assume it’s a weapon).

Controllers and the factory *do not* currently make use of the proxy upgrade pattern. Once a given vault is deployed it is immutable: Wildcat cannot change the logic of a deployed vault, and at no point has access to (or custody of) any assets within vaults beyond the fees incurred by the borrower. A borrower wishing to take advantage of any protocol upgrades made available via new controller or factory deployments will need to migrate to a new vault.

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



A controller contains:

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

Several controllers can exist concurrently, with different rules for permitting vault access and manipulating parameters, however in the current version of the protocol only one is deployed.

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

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

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

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

A vault is deployed with the following parameters:

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

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

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

- A vault with a capacity of 1,000 could only withdraw 900 maximum,
- Similarly, a vault that set its capacity to 1,111 to withdraw up to 999.9 could find itself delinquent at the end of the first withdrawal cycle if all interest was withdrawn.

- 1,200 allows for 1,080 to be withdrawn, but if a maximum of 1,000 is, then over a years worth of interest is waiting in the wings of the vault above and beyond the required reserves.

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

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

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

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

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

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

2.2.1 Borrower Adjusts Capacity

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

If a borrower decides to decrease the vault capacity, they are only able to drop it to a maximum of the currently outstanding balance (including interest). If a lender deposits 100 `WETH` into a vault and the borrower decides that actually, that'll do, they can't then drop the maximum capacity to 100, as there'll be a tiny fraction of interest that has already accrued in the intervening blocks.

2.2.2 Borrower Adjusts Vault APY

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

A borrower that *decreases* the vault rate will find themselves potentially having to return assets to the vault - the precise amount depends on the controller that deployed said vault - in the same transaction, with borrower withdrawals locked for a period of time thereafter. This is because the alternative is to enable a rugpull mechanic whereby a borrower could induce deposits at a high rate, then withdraw as much collateral as the liquidity ratio permits and drop the rate to reduce their future interest obligation.

We're not interested in permitting such behaviour, and we'd rather give the benefit of flexibility to lenders that decide that the new rate is no longer acceptable to them. We're aware that the relationship between APY and credit is not linear - in that a 20% decrease in APY may induce far more than 20% of deposited assets to exit - but by way of example, *if* the equation that determines how much collateral to return and temporarily freeze is -

$$Returnable = \left|\frac{NewRate - OldRate}{OldRate}\right| \times (Outstanding - Reserves)$$

- then dropping the vault APY from 5% to 3% in a brand new, fully subscribed 1,200 `WETH` capacity vault with a 10% liquidity ratio requires:

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

2.2.3 Borrower Allows Vault To Become Delinquent

In the event that a lender chooses to exit their position (either partially or fully) by redeeming their vault tokens and the borrower has withdrawn assets to capacity, it's likely that after the lender has left the scene that the vault will now be under its liquidity ratio. Using our aforementioned example:

1. Two lenders deposit into the vault: `A` lends 1,050 `WETH`, `B` lends 150,
2. Borrower withdraws 1,000 `WETH` and starts to use it (200 left),
3. Lender `B` decides to fully exit, queues a withdraw of 150 `WETH` plus interest,
4. The vault will be left with just shy of 50 `WETH` in reserves.

Given that the 10% liquidity ratio of the vault requires that it contains at least 105 `WETH` (10% of the remaining 1,050 `WETH` belonging to lender `A`), this vault will become delinquent at the end of the withdrawal cycle if lender `B` completes their withdrawal.

The purpose of the withdrawal cycle is to allow the borrower to handle circumstances such as this, depositing reserves back to the vault to ensure that no matter who chooses to exit, the minimum amount is maintained: in this example, the borrower has to return slightly over 55 `WETH` (to account for interest) before the (7 day, as specified by this vault) cycle ends. However, a borrower can elect - or forget - to not do this.

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

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

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

In the event that a vault continues to be delinquent and the grace tracker reaches six months, any lender with a currently active position in the vault have the *option* to mark the vault as having defaulted: a condition subsequent that terminates the legal contract underlying the vault and gives rise to grounds for a claim in debt against the borrower for the balances outstanding at that time.

2.2.4 Lenders Stampede For The Exit

It may be the case that due to market conditions (e.g. better rates offered elsewhere, news about the stability of the borrower, or myriad other sources), one or more lenders attempt to exit the vault for an amount greater than the current reserves in the same withdrawal cycle.

The fairest solution to this - assuming that the borrower does not deposit the amount requested back to the vault before the cycle is over - is to prorate the amounts that each lender is permitted to withdraw, relative to the amount that they are seeking to. To illustrate with our example vault:

- As before, we have two lenders: `A` lends 1,050 `WETH`, `B` lends 150,
- Borrower withdraws 1,000 `WETH`, leaving 200 in the vault,
- Lender `A` initiates a withdrawal cycle, requesting 300 `WETH`,
- In the same cycle, lender `B` requests 100 `WETH`,
- The result of the cycle depends on the actions of the borrower:
  - If the borrower deposits over 200 `WETH`, both lenders can fully withdraw, with the vault *potentially* becoming delinquent afterwards,
  - If the borrower deposits nothing, each lender will be permitted to withdraw half of their requested amounts (400 `WETH` requested versus 200 in reserves), and the vault *will* become delinquent.

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

Nonetheless, there is an option to specifically file a more 'formal' request to exit for a given amount on-chain via an event-emitting function. These have no binding power, but will be logged and displayed to all parties to a given vault.

2.2.5 Borrower Removes Access From An Active Lender

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

In the event that a borrower removes a lender from this list, this - unsurprisingly - bars that lender from depositing to any of their vaults. However, in the event that a lender has an existing active position in a vault, they are permitted to withdraw at their leisure: they cannot be 'forced' out unless the vault is terminated by the borrower, a process which we discuss in the next section.

2.2.6 Borrower Wishes To Terminate The Vault

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

After doing this, the ability for a borrower to withdraw assets is *permanently* frozen, and no further changes to *any* vault parameters can be made. It is then up to the lenders to withdraw their assets, although terminated vaults do have a public eject function available that forces a return of all assets to all lenders.

2.2.7 Borrower Loses Their Private Key

There's not really much we can do here. We don't want to enable the ability to update the borrower address for a vault, not least because it puts us in a position where we might be forced by a court to meddle, but because it also means all parties need to trust Wildcat to not interfere with an extant agreement.

If we *could* do this, someone with control of the appropriate archcontroller address could set the borrower of a fully-subscribed vault to a wallet they control, withdraw assets up to the liquidity ratio, and disappear.

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

Of course, claiming 'we've lost the keys' could be a feint by a borrower trying to evade repaying their loan - and if it appears that this is the case, the aforementioned option for a jilted lender to trigger a vault default when the grace tracker hits six months and initiate legal proceedings remains open to them.

2.2.8 Lenders Lose Their Private Keys

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

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

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

2.2.9 Lender Gets Placed On A Sanctions List

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

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

In the event that a lender is flagged in such a way, the sentinel can *elect* to call a 'nuke from orbit' function - which will require verifiable on-chain proof of the blacklisting of the target address before triggering - within any affected vault. The effect of this is to burn all vault tokens held by the lender, setting the vault state to one where the lender never deposited at all and effectively gifting the lender assets to the borrower.

The sentinel is a powerful tool, and not one that we expect to ever utilise barring another mass blacklisting event in the fashion of Tornado Cash (which was legal to utilise until it suddenly wasn't). We recommend both that candidate lenders take the lessons learned there to heart when determining the addresses they submit for vault access, and that borrowers consider this potential when going through their AML/KYC process for a candidate lender.

Our reasoning for the inclusion of the sentinel is founded in contract law - the legal agreement underpinning the vault may become frustrated if it becomes illegal to perform, and the effect of frustration is to release all parties from their obligations (notwithstanding statutory provisions, depending on which court system you're talking about). We would rather reserve the *right* to excise blacklisted addresses - innocent of wrongdoing as they might be - in order to preserve the independence and stability of the wider protocol.

Note also that the sentinel address falling into hostile hands will have limited impact on a vault beyond potentially requiring the vault to migrate to a new deployment: the *only* thing it can do is erase balances of flagged addresses, and it is highly doubtful that the same party that seizes control of a sentinel can also manipulate a Chainlink feed to add unflagged parties.

2.2.10 Borrower Gets Placed On A Sanctions List

If the sentinel detects that a *borrower* is added to a sanctions list, sorry, but it's just going to have to go straight to court per the terms of the underlying agreement. We aren't facilitating blacklisted assets getting distributed back to their original owners, given the often-strict liability of effecting asset transfers involving these addresses.

2.2.11 That's All She Wrote

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

One final point to raise here is the manner in which the protocol claims its fees. Where there are reserves in the vault, the fee recipient will simply initiate or enter a withdrawal cycle in the same manner that lenders do. In the case where a vault has insufficient reserves to cover the fees due, *vault tokens* will be minted to cover those fees that cannot be claimed directly from reserves.

A borrower is well-minded to note that these are interest-bearing and position the Wildcat protocol as a lender - our reasoning for doing this is to both incentivise a bare minimum of reserves to be maintained, and to grant the Wildcat protocol itself standing to participate in any legal proceedings relating to a claim in debt against the borrowers of a defaulted vault.

That's where things stand. We'll be live soon enough - there are a bunch of discussions with our legal team, code audits, and internal processes to get sorted.

3 Future Directions

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

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

3.1 Long-Dated Option Vaults

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

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

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

3.2 Credentialed Access

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

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

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

3.3 Interest In Different Tokens

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

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

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

4 Code Repository

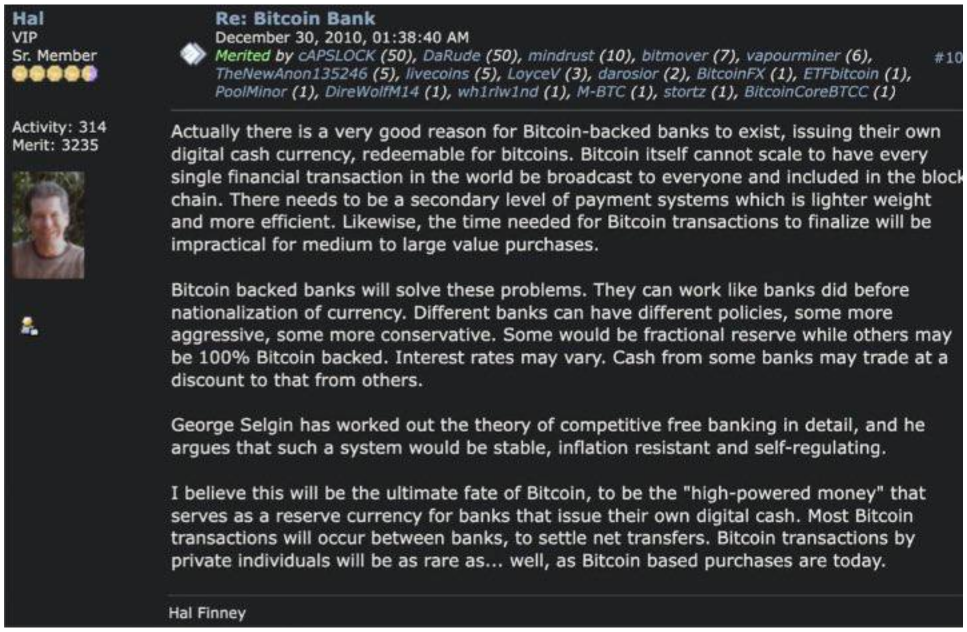
The code for the Wildcat protocol will be made available under the Commons Clause License, and can be accessed - along with the most recent version of this whitepaper and the technical specification - at:

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

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

5 One Final Point

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



Hal
VIP
Sr. Member

Activity: 314
Merit: 3235

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

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

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

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

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

Hal Finney

See you on-chain.

6 Changelog

The latest version of this whitepaper can be found at:

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

- Version 0.1 [15 May 2023] - initial draft