# ORIGINAL TEXT - EMAIL THREAD

**Thread Subject:** Analysis of full-RBF deployment methods

Hello Antoine, Thanks for taking the time to answer every email with detailed analysis! I can see it's a lot of work. I'll answer inline. wrote: and vaporware a Phoenix To solve the attack you described in [0], collaborative transaction protocols (such as dual-funded channels) need a \*reliable\* way to replace transactions. Otherwise, protocol parties using full-RBF may see replacements succeed in their own mempool, only to find out they weren't relayed to a miner once it's too late (ie. once the replacement that won is mined). I'm calling a full-RBF deployment reliable to the point at which any full-RBF-enabled node can broadcast a replacement and get it relayed all the way to a miner in a reliable manner (ie. with high-enough probability). Even if we deployed opt-out (or mandatory!) full-RBF now and miners adopted it immediately, it would take almost a year (assuming normal deployment times) for it to be sufficiently deployed in the relaying layer to be considered reliable. An opt-in full-RBF deployment, as currently proposed (ie. without 25600), has very little chance of getting us nowhere near that kind of adoption. Notice that 26323 (option 5 in the OP) has the advantage of getting us to a reliable full-RBF network the fastest (in particular, much faster than the current opt-in deployment) while not threatening zero-conf applications until the activation time. That is, 26323 gives us a way in which we don't need to choose between the security of one use case versus the other. We can have both. like be responsibility of business, rules already reviewing soft way I don't think asking for a predictable deployment timeline for a change that would put some applications at increased risk could be described as burdening the developers with solving every operational risk. This deployment method comparison's goal was precisely to soften the burden on core devs. Cheers, Dario

—

Hi Dario, Thanks for this analysis of full-RBF deployment methods! The subject was widely discussed at today Bitcoin Core IRC meetings: https://gnusha.org/bitcoin-core-dev/2022-10-20.log Personally, I still think deferring full-rbf deployment, while it sounds reasonable to let existing services and applications adapt their software and business models, doesn't come risk-free for the contracting protocols and multi-party applications affected by the pinning DoS vector. Deferring ad vitam aeternam left them exposed to disruptions when their traffic volume would start to be significant. While those use-cases (splicing/dual-channels/collaborative constructions) were mostly vaporware a year ago when I raised the issue, it turns out they have become a far more tangible reality today. Beyond the 3 coinjoins services (Wasabi/Joinmarket/Whirlpool), we have new things like ln-vortex, or Phoenix wallet and some LDK users planning to use dual-funded soon. I'm fully aware we still have community stakeholders opposing full-rbf on principle, I think this is highly valuable to exchange more perspectives on the subject, hopefully to harmonize understandings, and work altogether on creative solutions to accommodate as many use-cases as we can, in a compatible way. I don't know what the best full-rbf deployment method is, though one offering clear visibility and predictability to node operators, with a flag day activation like 26323 sounds like a reasonable one. As you said, I believe 6 months from now is far too short, maybe something like 12 months or 16 months (or more ?) is adequate. At least I fully rejoin on the risk of merchants centralization if we favor a timeline only favoring entities with sufficient manpower. I'm still looking forward to having more forums and communication channels between business/services operators and protocol developers, it sounds like functional responsibilities between protocol and application layers could be better clarified. However, I don't know if it should be the responsibility of developers to solve every operational risk encumbered by a Bitcoin business, like FX risk. I don't deny the interdependency between network policy rules and business risk, I'm just saying Bitcoin protocol developers have already heavily loaded engineering priorities between solving the half of dozen of Lightning vulnerabilities, working on the next consensus changes or reviewing modularity refactoring of Bitcoin Core to extend the feature set in a soft way (among tons of other examples). All speaking for myself, looking forward to having more perspective expressed! Best, Antoine

—

Hello list, Given that the release of 24.0 is upon us and there is little time to make a complex decision regarding the deployment method of full-RBF, we've documented the different alternatives and their trade-offs. I hope this helps get to the best possible deployment! Gist: https://gist.github.com/esneider/4eb16fcd959cb8c6b657c314442801ee Current deployment options 1. Antoine's PR 26305: leave 24.0 as is, and merge opt-out in 25.0 or later. 2. Marco's PR 26287: revert opt-in full-RBF in 24.0, and give more time to figure out what's next. 3. Marco's PR 26287 + Antoine's PR 26305: revert opt-in full-RBF in 24.0, and merge opt-out in 25.0 or later. 4. Marco's PR 26287 + Anthony's PR 26323 (just the date commitment): revert opt-in full-RBF in 24.0, and commit in 25.0 or later to a later date for opt-out activation. 5. Anthony's PR 26323: revert opt-in full-RBF in 24.0, and commit in 24.0 to a later date for opt-out activation. Notice that once full-RBF is fully deployed, having a config option to disable it is mostly a foot gun: you will only hurt yourself by missing some transactions. Maybe options 4 and 5 could remove the flag altogether instead of making it opt-out. There are a few more options, but I don't think they would reasonably have any consensus, so I trimmed them down to make it easier to process. Dimensions of analysis 1. Zero-conf apps immediately affected If we leave the flag for full-rbf in 24.0, zero-conf apps could be immediately affected. More specifically, as Anthony explained much more clearly [0], they would be in danger as soon as a relatively big mining pool operator enables the full-RBF flag. It turns out that the class of apps that could be immediately affected (ie. apps that were directly or indirectly relying on the first-seen policy in an adversarial setting) is larger than zero-conf apps, as exposed by Sergej [1]. Namely, the apps committing to an exchange rate before on-chain funds are sent/finalized would start offering a free(ish) american call option. 2. Predictable deployment date Committing to an activation date for full-rbf on the social layer (eg. "we'll merge the opt-out flag in 25.0") has the benefit of being flexible in the event of new data points but becomes less predictable (both for applications and for full-rbf proponents). Committing to an activation date for full-rbf on the code has the benefit that once node operators start deploying the code, the date is set in stone, and we can reason about when full-RBF will be fully deployed and usable. 3. Code complexity Handling the commitment to a date in the code introduces further code complexity. In particular, it's a deployment mechanism that, as far as I know, hasn't been tried before, so we should be careful. 4. Smooth deployment Full-RBF deployment has two distinct phases when analyzing the adoption in the transaction relaying layer. First, there will be multiple disjoint connected components of full-RBF nodes. Eventually, we'll get to a single(ish) connected component of full-RBF nodes. The first deployment phase is a bit chaotic and difficult to reason about: nobody can rely on full-RBF actually working; if it coincides with a high-fees scenario, we'll get a big mempool divergence event, causing many other issues and unreliability in the relaying and application layers. I'm calling smooth deployment to a deployment that minimizes the first phase, eg. by activating full-RBF simultaneously in as many transaction-relaying nodes as possible. 5. Time to figure out the right deployment Figuring out the right deployment method and timeline to activate full-rbf might be more time-consuming than what we are willing to wait for the stable release of 24.0. Decoupling the protection to zero-conf apps from choosing a deployment method and an activation date for opt-out might be a good idea. I'm probably forgetting some dimensions here, but it may be enough to grasp the trade-offs between the different approaches. Comparison Gist: https://gist.github.com/esneider/4eb16fcd959cb8c6b657c314442801eecomparison Timeline for full-RBF activation If we make some UX trade-offs, Muun can be production ready with the required changes in 6 months. Having more time to avoid those trade-offs would be preferable, but we can manage. The larger application ecosystem may need a bit more time since they might not have the advantage of having been working on the required changes for a while already. Ideally, there should be enough time to reach out to affected applications and let them make time to understand the impact, design solutions, implement them, and deploy them. Finally, if a smooth deployment (as previously defined) is desired, we can lock an activation date in the code and give relaying nodes enough time to upgrade before activation. Assuming that the adoption of future releases remains similar to previous ones [2], one release cycle should get us to 22% adoption, two release cycles to 61% adoption, and three release cycles to 79% adoption. Assuming a uniform adoption distribution, the probability of an 8-connection relaying node not being connected to any full-RBF node after one release cycle will be 0.14. After two cycles, it will be 0.00054, and after three cycles, it will be 0.0000038. Looking at these numbers, it would seem that a single release cycle will be too little time, but two release cycles may be enough. Cheers, Dario

## 

## DAVINCI- SUMMARY OF EACH EMAIL IN A THREAD (Approach-1)

**Email Subject:** Analysis of full-RBF deployment methods

**GENERATED TITLE & SUMMARY:**

**The Benefits of Opt-In vs. Opt-Out Full-RBF Deployment for Collaborative Transaction Protocols:**

Antoine replied to an email from Dario, who was discussing the need for a reliable way to replace transactions using full-RBF deployment. Dario suggested that opting in to full-RBF deployment, as currently proposed, has very little chance of achieving sufficient adoption. He then discussed option 5 in the OP, which is 26323, and stated that it would be the quickest way to reach a reliable full-RBF network while not threatening zero-conf applications until activation time. Finally, Dario clarified that his goal with the deployment method comparison was to reduce the burden on core devs.

**Exploring Full-RBF Deployment Methods for Bitcoin Core: A Discussion from Bitcoin Core IRC Meetings:**

Dario was thanked for his analysis of full-RBF deployment methods which was discussed at a Bitcoin Core IRC meeting. Antoine expressed that deferring full-RBF may not be risk-free as it leaves some contracting protocols and multi-party applications exposed to disruptions when their traffic volume increases. It is noted that these applications have become more tangible in the last year, such as coinjoins services, ln-vortex, Phoenix wallet and LDK users. Despite opposition to full-RBF on principle, Antoine is looking to harmonize understandings and work together on creative solutions. He suggested a flag day activation with a timeline of 12-16 months or more. Antoine is also looking forward to having more forums and communication channels between business/service operators and protocol developers to better clarify functional responsibilities.

**Analysis and Comparison of Full-RBF Deployment Options for 24.0 Release:**

Given the imminent release of 24.0 and limited time to decide on a deployment method for full-RBF, five options have been documented and their trade-offs analyzed. These options vary in terms of immediacy of effect on zero-conf apps, predictability of deployment date, code complexity and smoothness of deployment. There is also the consideration of giving time to figure out the right deployment. Muun can be ready with the necessary changes within 6 months, while the larger application ecosystem may need more time. A smooth deployment would require locking an activation date in the code and enough time for node operators to upgrade. Assuming similar adoption rates as previous releases, one release cycle could lead to 22% adoption, two cycles to 61% and three cycles to 79%. This suggests that two release cycles may be enough.

## ADA - SUMMARY OF EACH EMAIL IN A THREAD (Approach-2)

**Email Subject:** Analysis of full-RBF deployment methods

**GENERATED TITLE & SUMMARY:**

**23:**

Hello Antoine,Thanks for taking the time to answer every email with detailed analysis! I can see it's a lot of work. I'll answer inline. Conducting a deep look at how collaborative transactions are cultured from two previous articles in this series (virtual reality and data-driven design), we found that a way to replace transactions on a Phoenix-powered collaborative network would be to deploy a full-RBF deployment. This would mean that only transactions that are relay-enabled will be processed, and the memory of all the nodes that don't have relaying will be locked up. This will prevent the creation of replacement accounts andensralization of the blockchain.This is an opt-in full-RBF deployment, as currently proposed. It has the advantage of getting us to a reliable full-RBF network the fastest (in particular, much faster than the current opt-in deployment), while not threatening zero-conf applications until the activation time.We can have both a durable and reliable deployment method. Like be responsible of business, rules already reviewing soft way I don't think asking for a unpredictable deployment timeline for a change that would put some applications at increased risk could be described as burdening the developers with solving every operational risk.This deployment method comparison's goal was precisely to soften the burden on core devs. Cheers, Dario

**2x RBF on Bitcoin Core:**

The text below is a discussion of "full-RBF deployment methods" at a Bitcoin Core IRC meeting.Full-RBF deployment methods are discussed at a Bitcoin Core IRC meeting.

**Do you have any other recommendations?**:

New release 24.0 has a new flag to choose a different deployment method. Here is a list of different options:1. leaves 24.0 as is, and merge opt-out in 25.0 or later.2. revert opt-in full-RBF in 24.0, and merge opt-out in 25.0 or later.3. leaves 24.0 as is, and commit opt-out in 25.0 or later to a later date for activation.4. leaves 24.0 as is, and commit opt-out in 25.0 or later to a later date for activation.5. Anthony's PR 26323: revert opt-in full-RBF in 24.0, and commit in 24.0 to a later date for opt. activation.6. Anthony's PR 26323: revert opt-in full-RBF in 24.0, and commit in 24.0 to a later date for opt-out activation.7. Anthony's PR 26323: revert opt-in full-RBF in 24.0, and commit in 24.0 to a later date for opt-out activation.8. Marco's PR 26287: reverted opt-in full-RBF in 24.0, and merged opt-out in 25.0 or later.

## DAVINCI - SUMMARY OF ALL TEXT IN THREAD (Approach-3)

**GENERATED TITLE:** Analyzing Full-RBF Deployment Methods for Bitcoin Core with Antoine: Exploring Options, Communication and Timelines

**GENERATED SUMMARY:**

Welcome to this month's newsletter! This month, we’re discussing the need for a reliable way to replace transactions using full-RBF deployment. Dario suggested that opting in to full-RBF deployment, as currently proposed, has very little chance of achieving sufficient adoption. He then discussed option 5 in the OP, which is 26323, and stated that it would be the quickest way to reach a reliable full-RBF network while not threatening zero-conf applications until activation time.Antoine expressed that deferring full-RBF may not be risk-free as it leaves some contracting protocols and multi-party applications exposed to disruptions when their traffic volume increases. It is noted that these applications have become more tangible in the last year, such as coinjoins services, ln-vortex, Phoenix wallet and LDK users. Despite opposition to full-RBF on principle, Antoine is looking to harmonize understandings and work together on creative solutions. He suggested a flag day activation with a timeline of 12-16 months or more. Antoine is also looking forward to having more forums and communication channels between business/service operators and protocol developers to better clarify functional responsibilities. Given the imminent release of 24.0 and limited time to decide on a deployment method for full-RBF, five options have been documented and their trade-offs analyzed. These options vary in terms of immediacy of effect on zero-conf apps, predictability of deployment date, code complexity and smoothness of deployment. Muun can be ready with the necessary changes within 6 months, while the larger application ecosystem may need more time. A smooth deployment would require locking an activation date in the code and enough time for node operators to upgrade. Assuming similar adoption rates as previous releases, one release cycle could lead to 22% adoption, two cycles to 61% and three cycles to 79%. This suggests that two release cycles may be enough.We thank Dario and Antoine for their analyses of full-RBF deployment methods and are looking forward to working together on creative solutions. We will keep you updated on any new developments regarding this issue. Thank you for reading!

## ADA - SUMMARY OF ALL TEXT IN THREAD (Approach-4)

**GENERATED TITLE:** -Full-RBF -Activation date -Code complexity -Smoothing of deployment -L

**GENERATED SUMMARY:**

The article discusses the comparison of full-RBF and activation-time-based-adoption between the two types of networks, and how it is clear that the former would be more preferable when it comes to reducing Zero Conf applications. It also discusses the different adoption rates that are necessary for a network to be ready for release 24.0, and how this may be done in six months. Finally, the article discusses the question of when full-RBF will be activated at present time?

## 

# ORIGINAL TEXT - SINGLE EMAIL

Still trying to make sure I understand this concern, let me know if I get this all wrong. On 4/22/22 10:25 AM, Russell O'Connor via bitcoin dev wrote: Right, a key security assumption of the CTV-based vaults would be that you MUST NOT EVER withdraw more in one go than your hot wallet risk tolerance, but given that your attack isn't any worse than simply stealing the hot wallet key immediately after a withdraw. It does have the drawback that if you ever get a hot wallet key stole you have to rotate all of your CTV outputs and your CTV outputs must never be any larger than your hot wallet risk tolerance amount, both of which are somewhat frustrating limitations, but not security limitations, only practical ones. Of course I've always been in favor of a less constrained covenants design from day one for ten reasons, but that's a whole other rabbit hole :) Okay, Matt explained to me the intended application of CTV vaults off list, so I have a better understanding now. The CTV vault scheme is designed as an improvement over the traditional management of hot-wallets and cold-wallets. The CTV vault is logically on the "cold-side" and lets funds be sent from the "cold" side to \*one's own\* the hot wallet after the unvaulting delay. In this case, the hot wallet funds are always at risk, so it isn't unexpected that those funds could be stolen. After all, that is how hot wallets are today. The advantage is that funds can be moved from the "cold" side without needing to dig out the cold keys. The MES vault scheme applies to a different scenario. In the MES case it is the hot funds are inside the vault, and it is the hot key that unvaults the funds and sends them to \*customer's addresses\* after a delay. If the hot-key is used in any unauthorised way, then funds can be sent to the address of the cold key (the MES vault actually does something fancy in case of recovery, but it could be adapted to simply send funds to a cold wallet). The MES vault lie somewhere between "better" and "different" when compared to the CTV vault. If one is unwilling to use the MES vault on the hot side and have every withdrawl vetted, then, while you could use the MES design on the cold side like the CTV vault, it wouldn't really offer you any advantages over a CTV vault. However, if you are interested in managing all your payments through a vault (as I've been imagining) then the CTV vault comes across as ineffective when compared to an MES style vault.

# 

## DAVINCI OUTPUT

---------------------------

**GENERATED TITLE:** Exploring the Advantages of CTV and MES Vaults for Hot and Cold Wallet Management

**GENERATED SUMMARY:**

Russell O'Connor explained that the CTV vault scheme is designed as an improvement over the traditional management of hot-wallets and cold-wallets. It lets funds be sent from the "cold" side to one's own hot wallet after a delay. However, the MES vault scheme applies to a different scenario where the hot funds are inside the vault and it sends the funds to customer's addresses after a delay. If the hot-key is used in an unauthorised way, then funds can be sent to the address of the cold key. The MES vault offers more advantages than the CTV vault when managing all payments through a vault.

## ADA OUTPUT

---------------------------

**GENERATED TITLE:** The CTV-based vaults are designed to ensure that you cannot ever withdraw more than your hot wallet risk tolerance, and the MES

**GENERATED SUMMARY:**

Still trying to make sure I understand this concern, let me know if I get this all wrong. On 4/22/22 10:25 PM, Russell O'Connor wrote:The CTV vault scheme is designed as an improvement over the traditional management of hot-wallets and cold-wallets. The CTV vault is logically on the "cold-side" and lets funds be sent from the "cold" side to \*one's own\* the hot wallet after the unvaulting delay. In this case, the hot wallet funds are always at risk, so it isn't unexpected that those funds could be stolen. After all, that is how hot wallets are today. The advantage is that funds can be moved from the "cold" side without needing to dig out the cold keys. The MES vault scheme applies to a different scenario. In the MES case it is the hot funds are inside the vault, and it is the hot key that unvaults the funds and sends them to \*customer's addresses\* after a delay. If the hot-key is used in any unauthorised way, then funds can be sent to the address of the cold key (the MES vault actually does something fancy in case of recovery, but it could be adapted to simply send funds to a cold wallet). The MES vault lie somewhere between "better" and "different" when compared to the CTV vault. If one is unwilling to use the MES vault on the hot side and have every withdrawl vetted, then, while you could use the MES design on the cold side like the CTV vault, it doesn't offer you any advantages over a CTV vault. However, if you are interested in managing all your payments through a vault (as I've been imagining) then the CTV vault comes across as ineffective when compared to an MES style vault.