AUTHORIZED_FETCH

artistmarciax

Milestone C

GoToSocial Research

Introduction

From 2019-2021, I was a moderator on the Black-created Mastodon instance playvicious.social. In the beginning, the level of inter-user conflict and trolling targeting PlayVicious was not dissimilar from any other social media: it was present, but not overwhelming. As we began to shift our discussion on the instance away from racism from white people, to include intra communal issues and anti-blackness from non-black communities of color, PlayVicious (PV) began to face a tide of targeted harassment: screenshots of followers-only posts being shared to private Discord servers, anti-black harassment, stalking and more.

Despite my instance suspending another instance that was targeting myself and fellow users, users from the suspended instance were still able to see, share, and comment on our posts. Those interactions were not visible to members of playvicious.social, and therefore seemed to be happening behind our backs.

As a moderator, it was not clear to me how the software was handling suspension/defederation, and why suspended instances were still able to interact with our posts. From the perspective of a suspended instance, it may have seemed like we'd never suspended them at all. This was counterintuitive: in a common-sense understanding of instance suspension, if I suspend another instance I am ensuring that we no longer have access to one another in either direction (them to us, us to them). But to my dismay this was not the case.

In March 2020, in the hope of better understanding the situation, I posted the following:

> "Here's what I don't understand. In the current window opened, I can't see toots from a certain instance, but I know for a fact they are rb'ing toots from

> members of this instance…What's the point in suspension if the boundary is totally avoidable?..."

Through responses to my question, I realized (along with other PV users), that in fact, instance suspension just means that the suspending instance (instance A) is blocked from seeing content on the suspended instance (instance B). Instance B, however, can very well still see, boost, like, and engage with posts from instance A. This meant that in order for two instances to be wholly cut off from one another, they would have to engage in a mutual block (instance A suspends instance B, instance B *also* suspends instance A). After more posting and discussion in the thread, other people started to make this same realization too.

Because of this gap in the tooling, and the lack of communication about this gap, PV was unable to mitigate cyber stalking and harassment effectively, and a flourishing and diverse community had to shut itself down.

Authorized fetch / Secure mode

If a fediverse moderator/admin does not have a tech background, and is not in constant communication with developers, or checking in on updates to the software their instance uses, they may be out of the loop as to what tools are available, and what the available tooling actually does. For moderators who do not 'speak' or center tech in their lives, and are more interested in community building than technology, it can be difficult to stay up to date with changes that are being made to the code. This has a negative impact on the moderator's trust in the tooling provided, and, by extension, their ability to moderate effectively.

This is illustrated by what occurred with PlayVicious, where we were unable to stem the tide of harassment using the provided tools. What we did not know at the time, but only learned about much later, is that there *was* in fact a tool that would have helped: authorized_fetch, also referred to as secure mode.

As explained on the website fedi.tips, authorized_fetch

> "makes user blocks more effective, as it makes it harder for blocked people on other servers to interact with public posts from people who blocked them. (It only really helps with public posts, private posts are already protected against trolls.)" (https://fedi.tips/authorized-fetch/)

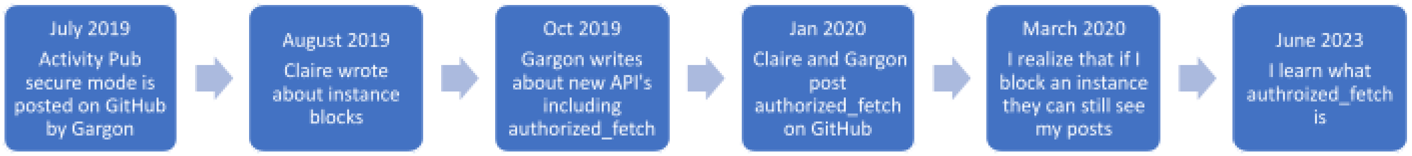
If users at PV knew that instance blocks were not mutual and that bad actors were still able to observe our timelines and interact with our posts, we would have then moved to making more informed choices about our posting visibility as individuals. In other words, followers-only posts or local timeline posts. Instead, we made what we believed to be a safe choice but in fact our timelines were being constantly monitored. For example, if I were to post about a topic regarding racism or colorism, “blocked” users could still see these posts, write about it on their timelines (which I could not see) and then other users would come back and harass me for sub-posting about other people even though I was writing about broader topics. This dynamic of myself and others at PV not being in control or in the know of who could see our posts created a situation ripe for rumors, drama and serious miscommunications.

The original set up for instance blocking was done in a counter productive way against user safety. As a user you're implicitly consenting to certain people seeing your post when you write a post, that's part of the whole model of social media and it's something that most people sort of understand by now. This implicit consent occurs because this model for posting has been around for so long, but in the absence of clear messaging around exactly *who* might see your post, you're working with insufficient information to make a clear consent decision. Especially if blocking on established social media sites has always been for blocking the person from seeing your own posts, not blocking yourself from seeing theirs. Therefore, as users/mods/admins, not knowing about the initial function of a block and moreover, no clear communication about the development and release of authorized_fetch in 2019, meant that for a couple of years, users have not been able to informed decisions and actively consent to who can and cannot see their posts.

Data

This section is an overview of collected data regarding authorized_fetch. First, I provide a timeline that shows the discussion of the block function, then authorized_fetch, and then the public posting from the developers and the release of authorized_fetch.

The questionnaire was for admins/mods that are familiar with authorized_fetch and its function, or for those who are not and learned about authorized_fetch through seeing this questionnaire be posted on the fediverse.


July 2019 Activity Pub secure mode is posted on GitHub by Gargon
August 2019 Claire wrote about instance blocks
Oct 2019 Gargon writes about new API's including authorized_fetch
Jan 2020 Claire and Gargon post authorized_fetch on GitHub
March 2020 I realize that if I block an instance they can still see my posts
June 2023 I learn what authroized_fetch is

July 2019- https://github.com/mastodon/mastodon/pull/11269

Aug 2019- https://social.sitedethib.com/@Thib/102608855195132093

Oct 2019- https://blog.joinmastodon.org/2019/10/mastodon-3.0-in-depth/

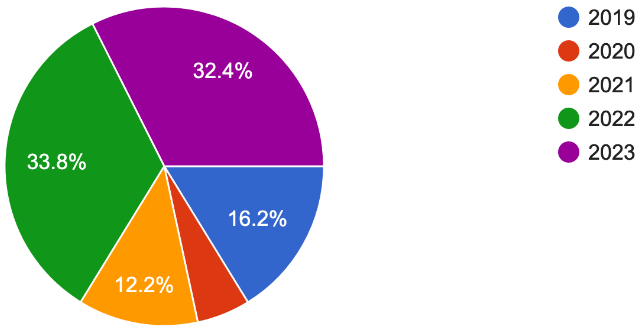
Jan 2020- https://github.com/mastodon/documentation/commit/ae93e4b66d6f4d2a66c4788b83d2e6c66f47c01a

March 2020- I learn blocking an instance doesn’t constitute a mutual block

June 2023- I learn in full what authorized_fetch is

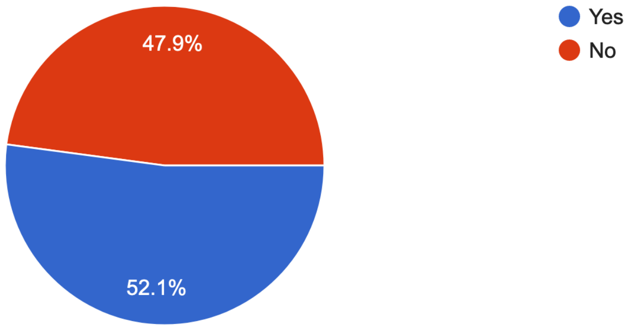
When did you learn about authorized fetch?

74 responses



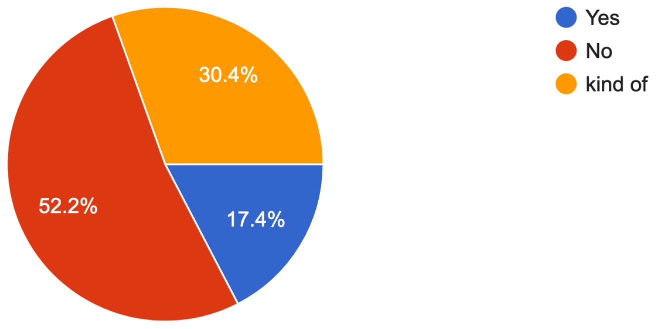
Did you understand what authorized fetch meant (its function) when it was first discussed?

71 responses



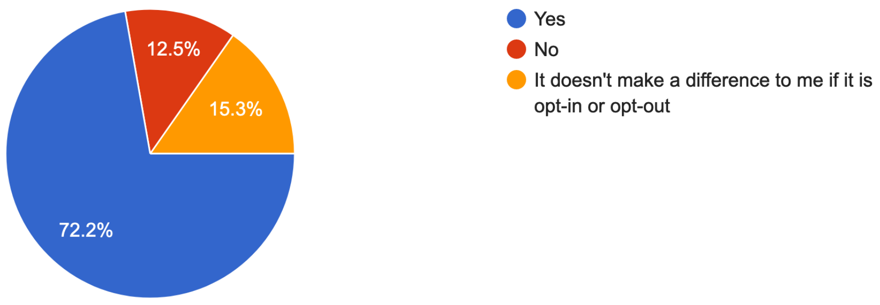
As a mod/admin, were you made aware of the availability of authorized fetch in a clear manner?

69 responses



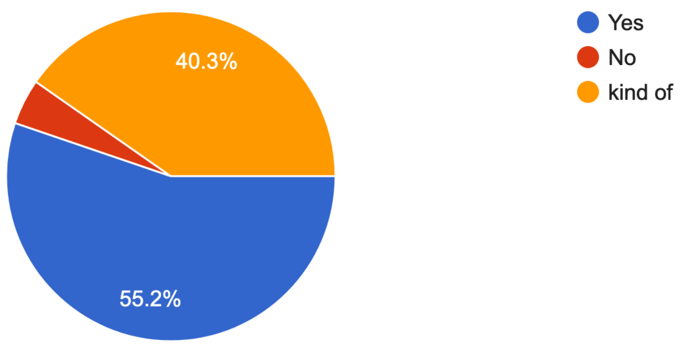
Do you wish authorized fetch was opt-out versus its opt-in?

72 responses



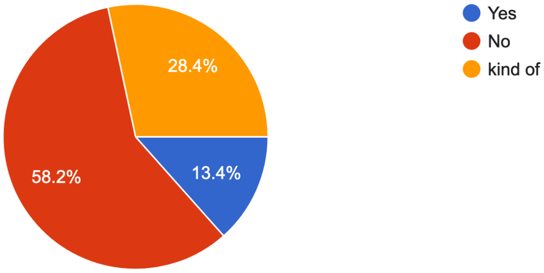
Do you think that authorized fetch is a useful tool and easy to implement?

67 responses



Does implementing authorized fetch create problems for you in regards to the code/software working properly?

67 responses



Conclusion

Prior to the research and writing of this piece, I was under the impression that authorized_fetch was a fairly new tool, hence why myself and many other users that I speak to on the fediverse were unaware of what authorized_fetch was. However, after beginning to speak to more instance admins, I learned that the feature was in fact a few years old. The lack of a proper two-way block, and its solution of authorized_fetch being kept below the radar and not widely known to admins, mods, and users helped to sustain a hostile environment, particularly for queer people of color. The small pool of data that I collected demonstrates that many admins were not made aware of authorized_fetch until years following its release, and a few people responded that they still did not know what it is at all, or only learned of it because of my questionnaire.

Authorized_fetch should have been better advertised and explained so that everyone can actually understand what it does and why it is necessary and beneficial to user experience. Despite my own inability to code, my understanding is that there are instances that do not use it because it can create communication issues between servers. This is a good example of why such safety tools need to be implemented and consistently communicated from the very start. User safety is of the utmost importance, and our ability to control and curate our experience on social media is part of what makes the fediverse enticing and important. The history of authorized_fetch exemplifies how the can fail users, but also shows how important communication from developers to users can be with regard to the tools users need to maintain safety.

References

https://designjustice.mitpress.mit.edu/pub/cfohnud7/release/4

https://www.artistmarciax.com/post/archivedcontentpost. (April 2022)

https://social.sitedethib.com/@Thib/102608855195132093

https://blog.joinmastodon.org/2019/10/mastodon-3.0-in-depth/

https://github.com/mastodon/documentation/commit/ae93e4b66d6f4d2a66c4788b83d2e6c66f47c01a

https://fedi.tips/authorized-fetch/

https://oliphant.social/@oliphant/110617039895509438