Parental controls in GNOME
Philip Withnall

>> Hey, everyone. This will be the talk about parental controls. If the moderator could give me presenter rights, I will start to get the presentation set up.

>> Hello. And welcome to GUADEC. A few things I would like to mention before we start. You can check the whole schedule for the talks today at GUADEC.org. You can find the Etherpads of the sured notes not BigBlueButton room to ask about the presentation and also the live captioning links. The social events for tonight. We have a 21 UTC. We'll host a cooking class. You can check the ingredients at the website. At 22 UTC, Molly will host a social event, LGBTQ + party. And then the social hour hosted by Matthew Miller.

All social events will happen here at the Track 1. Now we'll have Philip Withnall present on parental controls in GNOME. Welcome.

Philip Withnall: Hi, everyone. And thanks for the announcements and introduction. Hopefully everyone can hear and see me. Just check the chat. Richard --

>> We can hear you and see you. The chat is no longer here on the BigBlueButton. There is a link in the shared notes where we can talk about this in Rocket Chat. The link is in the shared notes.

Philip Withnall: Cool. Thanks. And we've got a couple of minutes so I'll wait 2 minutes until the scheduled time. If that makes sense.

Okay. Seems like people are have stopped joining. So, let's get started. And so, this is an introduction to the parental controls feature. Which was covered in GUADEC last year. Covered from a user and technical perspective. And then look at how to integrate with parental controls in your app. And finish with some coverage of future plans.

And next slide. So, I have been working on this for a few years. And one of the consistent pieces of feedback we got from users and integrators was that they needed support for parental controls. Which is the ability to limit what users can do on the system. Particularly relating to access to and installation content like websites or new apps.

With the growth of Flathub, a wide variety of apps are now available for users to install. And some of which parents might think are not appropriate for their kids. So, we introduced the first version of this, of parental controls, to try things out and then looked at upstreaming it in hack fest in London in March of 2019. Which I write about on my blog.

That was a productive Hackfest. And it was being from a variety of people and distros, and the related feature of digital well-being. That's applying limitations not to someone else, but to yourself on your computer. It's not wasting time or staying on the computer too long without a break or working too late and so on.

And currently, parental controls support restricting access to specific applications chosen by the administer with web browsers having their own special switch. The administer can restricting additional applications. Highlight it. And allow them to install applications which are suitable for certain age ranges. Restrictions on application and installation currently only work with Flatpak apps. That can be applied separately to each non-admin user. Two tabs at the top for my friend Bob and my friend Test. And they can only be changed with admin privileges.

This is the separate link because it likely supports statistics and other non-settings like content in the future. There are various integration points of parental controls. The parental control itself, the settings and widgets and lush was in the libmal GenContent. Flatpak keeps them from them being installed, and disallowed for being launched. And software is the same. And the control center hides them from the apps panel like in the area where you can set where notifications are allowed and it has a link through the parental controls settings so you can usually find those to set it up. And initial-setup, the most recent integration. for the parents and child uses to be created at app setup time. And allows you to set up the initial restrictions on the child account.

So, here's an example of what we've put into initial setup. The -- the new bit here is this tick box. If you don't tick that box, then initial setup happens as normal as it did before. But if you do, you tick the box, that sets up parental controls for this user. Then the user you create here becomes a new child account and it doesn't have administrative privileges. So, you tick box and continue. You can then set up the parental controls for that new child user using the same settings as I showed on the previous screenshot. And then the next step allows you to set a pass word for a separate parent account. Which is created at the end of the setup as well. So, you end up with two users. You end up with a child without administrative privileges, the parent with administrative privileges. And the child can't control what the parent controls because they don't have access to that. And once you finish setting the parent password, then you proceed as before.

So, parental controls integration in the initial setup was initially released in EOS 3.8, the beginning of 2020. We included anonymous statistics reporting to measure the uptake of parental controls. And we have a plot of it here. There's not much data yet because the start of May was not that long ago. And a lot of strange world events have happened in the meantime. That interpreting the graph, you can see it seems consistently that 2% of new users have been enabling parental controls during initial setup. And the curve is pretty straight.

Obviously, that only includes users who have opted into statistics reporting and only includes users who have installed from scratch since 3.8 was released. Users from previous release are not getting these stats because they didn't do initial setup again.

So, how do parental controls work? This diagram shows an example of how they're implemented in Flatpak to potentially disallow the user from installing an app that's not age appropriate. Say the user wanted to install WolfenDoom? It's a violent game and there is the metadata for WolfenDoom from the app stream file to libmalcontent, the library. And it describes the app. Whether it contains bloody violence, religious references, sexual content. That kind of thing. Each app, not just games, should have this data. Otherwise it could be maximally violent, sexual and full of gambling and so on. We wouldn't have the data to say otherwise.

And libmalcontent queries the settings. Which indicates how much violence and gambling the user is allowed to see. Those are returned and compares those settings against the apps and the data to work out whether the installation is allowed. In this case, we decide that the user is not allowed to install WolfenDoom because it's too violate. Flatpak will query polkit if they want to override malcontent. The parent may decide it's okay to use that app. Or may want to bring up their child differently from the generic age restrictions that we have. So, it will pop up an authorization dialogue. And if the user enters the admin password, selection is allowed. If it's not, then it's denied.

This is all implemented within the same user process space. So, it's not actually real security. The user will always find a way around it. For example, by downloading a version of WolfenDoom outside of Flatpak or a myriad of other ways. But this feature should prevent the average child from doing things that they're not supposed to do.

While restricting the installation of apps is implemented using metadata. Restricting access to apps which are already installed use blocklist, a bit simpler. Allows the administer to have age-inappropriate apps installed for all users, but restrict them on a user-by-user basis. If there's two parents and a older and younger child account, you could have WolfenDoom for the whole family except for the young kid to enjoy.

Like all data for parental controls, the blocklist is stored in account service which has this sidecar database for storing information about user accounts.

So, how do you integrate your app with parental controls? And if you do nothing else, you should please spend 5 minutes doing this. Which is to add all the metadata to the application. Put it in the AppStream file. No gambling, no online chats, anything like that. The metadata needs to say that. Otherwise we just don't know. There's an easy online questionnaire on Richard's GitHub page that generates the right metadata after a series of questions and you copy and paste that into your AppData file. Here is an example of it.

Note the -- the attributes are not all out violence. That's stuff I used before. There's stuff to do with chat, sharing social info, location, contacts, things to do with gambling and in-app purchases. And there are a variety of other attributes which are documented on the OARS website. If any OARS data is provided. If you provide this content rating element in your app data, then missing attributes are assumed to have the value none. The least intense value. If no data is provided, so if you don't provide the content rating at all, then they're all seen to have the intense value. It's assumed that your app is absolutely full of violence and gambling and chat and everything because we don't have the info otherwise.

If your application has more complex parental controls needs, for example, it might display various kinds of content dynamically. Might be an encyclopedia or something -- a web browser. And you feel you need to restrict some of that content for some users but not others without having it as binary as is this app allowed in its entirety for the user or not? You might want to use libmalcontent in your app to filter on a case-by-case base.

There is sample code in the malcontent repository here. And get in touch and I can help out because it's not something anyone has done yet. But the system is designed for it. So, should be fairly easy to implement.

And as far as I know, currently, and OARS is the only with parental controls. Enabling it should be enabling the libmalcontent dependency in the apps earlier, and ensuring that all the things you ship have the correct metadata. If you allow from other than Flatpak, distribution repositories, that kind of thing, or if you have your own Flatpak repositories that aren't Flathub, you may need to hook up AppStream metadata from your package manager to allow GNOME-software to have access to that data. Because without it, it can't know and it can't do the filtering. Flatpak uses the same metadata as the OARS software. That is, again, something that distributions might need to hook up and add support for. Please get in touch with me if you've got questions about it.

But the main idea is that distributions might want to be maybe moving towards moving Flatpak for these kids of apps. Maybe this is a problem that will just go away with time.

And the Hackfest in March 2019, various other parts of the feature were discussed. And digital wellbeing was enthusiastically received by people. I would like to work on that next. Screen time in particular. It should be relatively easy to implement. The idea is that it limitings the amount of time you can spend on the computer without a break. It's your little reminder. Little interruption to say, maybe you should take a break now. If anyone is keen to work on it, I'm happy to mentor and provide support if I don't get there myself first. So, we've got mockups here for how screen time functionality could work. Put together by Allan Day nine months ago, I think. And that's the general idea where we're looking at to go next. This is the screen time app in the control center.

And the eventual idea is that you get all of this functionality. But we can implement it in stages. Screen time, there. Give an indication of when to break. And how long chunks of time you should be using the computer for. And then there will be some breakdowns of stats on how long you use computer each day. And maximal time that you're allowed to use the computer for on a single day. That kind of thing.

The functionality for that could be common between screen time, digital wellbeing, between parental controls because there's also this idea that parents might want to limit the amount of time that their children spend on the computer. Or, again, limit it to an hour a day or whatever. And so, there's a lot of common infrastructure that could be shared between those things.

And that's it. A bit of a rapid overview of parental controls. And where we're at. And so, if anyone's got any questions. Let's see what we've got. So, the first question, the gnome-initial-setup part merged upstream? Yes, it is. It got merged a few months ago. It's disabled by default. Controlled by configuration option. But, yeah. Just turn it on at the malcontent dependency and it should work.

The second question, the parental controlled account should be different from the system account, maybe the parent one? I'm not entirely sure I understand this. But I guess this is something to do with how the accounts differ. So, when you're setting parental controls on accounts, you can only set them on non-admin account. The child account, the terminology we're using, but, yes, calling it the parentally controlled account. Yes, it has to be separate from the main account because it can't be an administer. It can't have pseudo rights. Because the child could enter their own password and authorize parental controls. When you have parental controls enabled, you have to have at least two accounts. One parent and one child. Third question. Could the screen time functionality integrate or borrow from the existing tool Workrave? I've never looked at Workrave, but possibly, yes. It depends on how we design the integration to the user's session.

I was imagining that it would be something that would return to the shell which would maybe dim the screen and pop up a notification. So, it's a bit more of a whole desktop experience than just a notification popping up. Something that's a bit like how the night time color change is implemented at the moment. Where it's not just one bit of the screen changing on it. It's the whole thing. But we can certainly look at that and see if those codes could be reused or we could reuse in a new type project.

Fourth question. The screen time on iOS shows applications that cannot be launched after a certain time by shading their icon. Can the Shell change the appearance of non-launchable apps by using libmalcontent? Sure, if it's non-launchable, it will hide if the icon and pretend it doesn't exist. We decided that made more sense than shading or hiding them for child users. Because that would kind of give the appearance of like forbidden fruit. There's this game that looks quite fun and quite interesting because it involves shooting lots of things and you can't launch it. It's the temptation of maybe I could just fiddle around to launch it. Or maybe I could find a different way of launching it. And it seems better to not have it presented at all. But from a screen time point of view, you're trying to nudge the user into behaving better or having more -- many sustainable habits, then, yeah. We could -- we could shade the icons or something or make them less attractive to run. And potentially that. So, libmalcontent provides an API that says is this on or not? And the shell would use that to render things differently or hide things as appropriate.

Fifth question I think is not very serious. I do have a recipe for flapjacks. But I'm not gonna share it. Because it's mine. And the sixth question, still being typed out. Am I familiar with certain project which does screen time? No, I'm not. But thank you for the link. Again, we'll look at that. That's -- the answer for that, I guess, is fairly similar to the previous one about Workrave. Where we can take inspiration from those things or maybe reuse bits of them or the whole thing. The idea would be to come up with a design-driven solution that fits in well with the whole desktop. And if code exists to do that already, then that's great.

But maybe some new code will need to be written. And that's it for the questions. Does anybody have any other questions that you want to add? No. Cool. Well, I guess that's that. Thank you very much for your attention. And drop me a line by email or on Rocket Chat if you want to discuss things afterwards.

>> Thank you for this talk. We still have some time left before the next talk. I think the next is in 10 minutes. Perhaps we should wait for the schedule.