USN-4764-1: GLib vulnerability

It was discovered that GLib incorrectly handled certain symlinks when
replacing files. If a user or automated system were tricked into extracting
a specially crafted file with File Roller, a remote attacker could possibly
create files outside of the intended directory.

How To: Enable Numlock on Modern Lubuntu

At the time of writing, Lubuntu 18.04 is soon to be no longer supported. With the change to LXQt, many of the old tweaks no longer work and you'll need to learn some new ways to do things. In this article, I'll explain how to put numlock into its on position automatically with Lubuntu 20.04 and up.

First, we're going to install 'numlockx' and that's pretty easy. Start by using your keyboard to open the terminal with CTRL + ALT + T. (I really like that keyboard indicating layout feature!)

Now that you have that open, try the following:

[code]sudo apt install numlockx[/code]

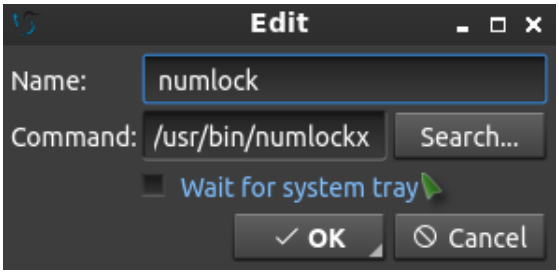
Next, open your applications menu, click on Preferences, click on LXQt settings, and then click on Session settings.

Once you have that open, click on Autostart (on the left) and

then Add (on the right). Once that is open, give our numlock a name and enter the following command:

[code]/usr/bin/numlockx[/code]

(Don't forget to click Ok to save it.) It should look a little like this:



See? This is a nice easy one!

And, that's it. Reboot and it should work. I can't really drag this out to making it a longer article. Sure, there are other ways to add something to autostart, but I see no reason to do this particular exercise any other way. Why add complexity when this just works out of the box?

Like always, thanks for reading. Don't forget to sign up for the newsletter. Chances are that I'll eventually cease posting at all the various forums (when the pandemic is over) so this will help you keep up with the site and keep in touch. Don't worry, I won't send you any spam.

USN-4754-3: Python vulnerabilities

USN-4754-1 fixed vulnerabilities in Python. This update provides

the corresponding updates for Ubuntu 18.04 ESM and Ubuntu 20.04 ESM.
In the case of Python 2.7 for 20.04 ESM, these additional fixes are included:

It was dicovered that Python allowed remote attackers to cause a denial of
service (resource consumption) via a ZIP bomb. (CVE-2019-9674)

It was discovered that Python had potentially misleading information about
whether sorting occurs. This fix updates the documentation about it.
(CVE-2019-17514)

It was discovered that Python incorrectly handled certain TAR archives.
An attacker could possibly use this issue to cause a denial of service.
(CVE-2019-20907)

It was discovered that Python allowed an HTTP server to conduct Regular
Expression Denial of Service (ReDoS) attacks against a client because of
urllib.request.AbstractBasicAuthHandler catastrophic backtracking.
(CVE-2020-8492)

It was discovered that Python allowed CRLF injection if the attacker controls
the HTTP request method, as demonstrated by inserting CR and LF control
characters in the first argument of HTTPConnection.request.
(CVE-2020-26116)

Original advisory details:

It was discovered that Python incorrectly handled certain

inputs.
An attacker could possibly use this issue to execute arbitrary code
or cause a denial of service. (CVE-2020-27619, CVE-2021-3177)

How To: Properly Install Proprietary Drivers in Ubuntu

There is some confusion about installing the proprietary drivers in Ubuntu. This article hopes to clear that up by telling you how to properly install drivers in Ubuntu.

First, this only works for the drivers that Ubuntu has access to. In this case, it’s usually things like graphics cards, sound cards, some networking gear, and things like that.

Ubuntu does not have all the possible drivers. If you have to go get them from GitHub and compile them yourself, this obviously isn’t the article for you.

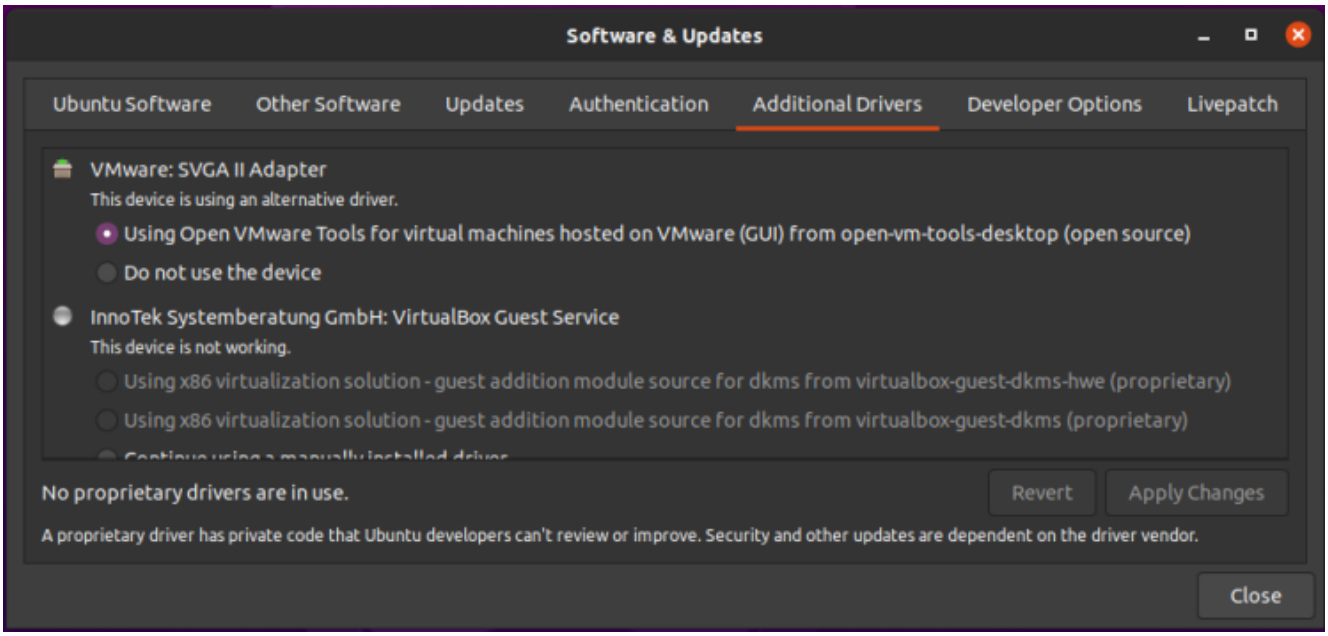
One of the most confusing is the Nvidia video card drivers. If you use Ubuntu, an official flavor of Ubuntu, or a derivative of Ubuntu, **DO NOT** download the .run file from Nvidia’s site. While it may work, it will quite possibly not work with dkms and you will have to spend significant time fixing it every time the kernel is updated. It quite likely lead to breakage.

Yes, this means having some patience. But, have some patience because the drivers *will* make it down to the repos and will have then been tested. The drivers you get from the official repos will not only update, they’ll update with the rest of the system **AND** they’ll work properly with the kernel updates. When the kernel updates, the system itself will insert the

appropriate drivers calls.

Doing this any other way will quite likely lead to hardship – and it's a hardship that you don't need to have. It's a hardship that's easily avoided. If you read the forums and question/answer sites, doing this the wrong way results in at least one question *almost every single day*.

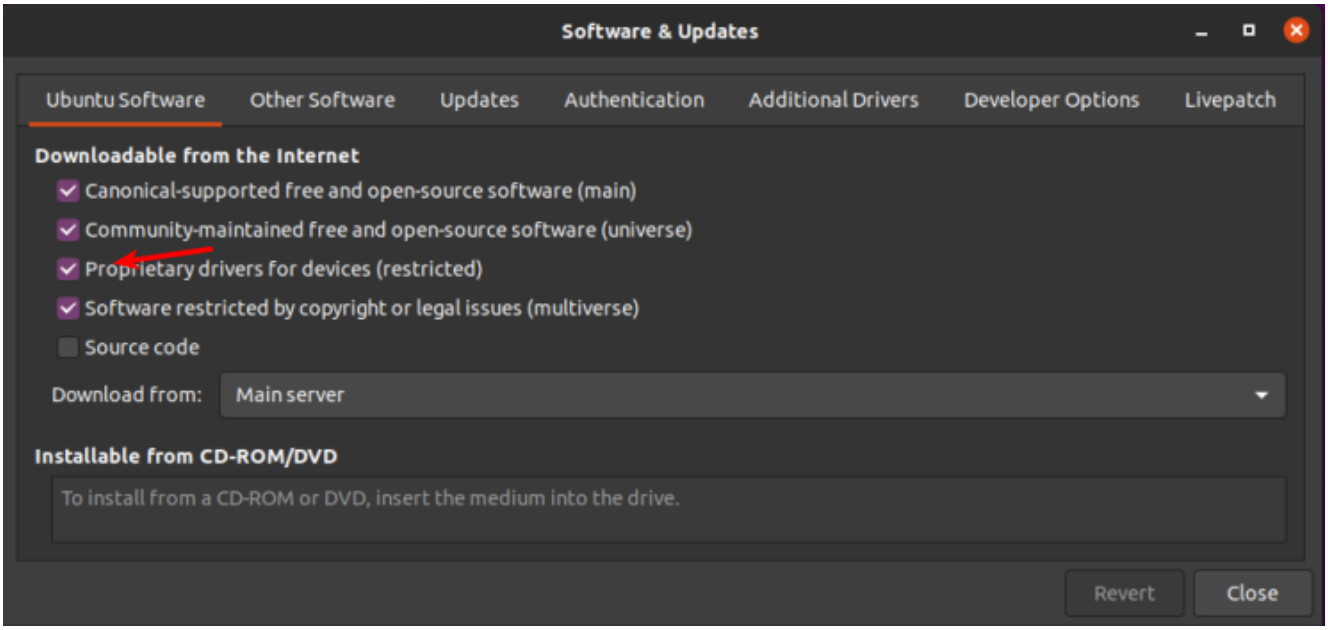
The first way is easy. It's even in the GUI. Search your menu for 'Additional Drivers' or similar (it may be only listed as "Software & Updates" depending on your Ubuntu flavor). It looks like this:



This should be self-explanatory, so I will just leave this here.

See? Pretty easy. Just pick what you want, apply the changes, and reboot.

NOTE: You will need to have the 'restricted' repository enabled in order to do this. That should be assumed, but some of you may not know this. So, a quick screenshot should make this even easier to figure out – it's in the same app as the above screenshot, but it's in the first tab. It looks a little like this:



The arrow should make it clear. That repo needs to be enabled for this.

The second way is to use the terminal. Let's go ahead and get the terminal opened by pressing CTRL + ALT + T on your keyboard.

Now, let's check and see what drivers we can automatically install from the terminal:

[code]ubuntu-drivers devices[/code]

The output should let you know what drivers are available for your devices. Again, this is pretty self-explanatory. You really don't even need to enter that command, you can do it all automatically and get the recommended drivers automatically installed. It's easy. Just run this:

[code]sudo ubuntu-drivers autoinstall[/code]

That will go through and install the recommended drivers for your devices automatically. That will install the proprietary drivers, the ones with binary blobs and decidedly not opensource drivers. If those are the drivers you want, that's the easiest way to install them.

Simply run the command, reboot, and you're done. Not only are

you done, you shouldn’t have to mess with them again – ever again. They will update automatically, they will automatically be applied when you update the kernel, and they will *generally* just work.

For the time being, we’re ignoring the idea of using the opensource drivers. I’ll simply say that I do quite well without needing proprietary drivers. I hardly ever bother installing them – unless I absolutely need a feature that’s not offered with the opensource drivers. I find that it’s still a working operating system and I can still easily meet my goals. You do you and you make the decision, but at least do it the right way after making that decision.

As always, thanks for reading. There’s a newsletter that will email you when a new article is published. That’s all it does. If you’re wanting to keep up with the site, that’s exactly how you do it! Well, there are push notifications available for those that prefer that. So, you do have choices! Either way, I won’t send you any spam. I promise!

How open source communities are driving 5G’s future, even within a large government like the US

In mid-February, the Linux Foundation announced it had signed a collaboration agreement with the Defense Advanced Research Projects Agency (DARPA), enabling US Government suppliers to collaborate on a common open source platform that will enable the adoption of 5G wireless and edge technologies by the

government. Governments face similar issues to enterprise end-users – if all their suppliers deliver incompatible solutions, the integration burden escalates exponentially.

The first collaboration, Open Programmable Secure 5G (OPS-5G), currently in the formative stages, will be used to create open source software and systems enabling end-to-end 5G and follow-on mobile networks.

The road to open source influencing 5G: The First, Second, and Third Waves of Open Source

If we examine the history of open source, it is informative to observe it from the perspective of evolutionary waves. Many open-source projects began as single technical projects, with specific objectives, such as building an operating system kernel or an application. This isolated, single project approach can be viewed as the first wave of open source.

We can view the second wave of open source as creating platforms seeking to address a broad horizontal solution, such as a cloud or networking stack or a machine learning and data platform.

The third wave of open source collaboration goes beyond isolated projects and integrates them for a common platform for a specific industry vertical. Additionally, the third wave often focuses on reducing fragmentation – you commonly will see a conformance program or a specification or standard that anyone in the industry can cite in procurement contracts.

Industry conformance becomes important as specific solutions are taken to market and how cross-industry solutions are being built – especially now that we have technologies requiring cross-industry interaction, such as end-to-end 5G, the edge, or even cloud-native applications and environments that span

any industry vertical.

The third wave of open source also seeks to provide comprehensive end-to-end solutions for enterprises and verticals, large institutional organizations, and government agencies. In this case, the community of government suppliers will be building an open source 5G stack used in enterprise networking applications. The end-to-end open source integration and collaboration supported by commercial investment with innovative products, services, and solutions accelerate the technology adoption and transformation.

Why DARPA chose to partner with the Linux Foundation

DARPA at the US Department of Defense has tens of thousands of contractors supplying networking solutions for government facilities and remote locations. However, it doesn't want dozens, hundreds, or thousands of unique and incompatible hardware and software solutions originating from its large contractor and supplier ecosystem. Instead, it desires a portable and open access standard to provide transparency to enable advanced software tools and systems to be applied to a common code base various groups in the government could build on. The goal is to have a common framework that decouples hardware and software requirements and enabling adoption by more groups within the government.

Naturally, as a large end-user, the government wants its suppliers to focus on delivering secure solutions. A common framework can ideally decrease the security complexity versus having disparate, fragmented systems.

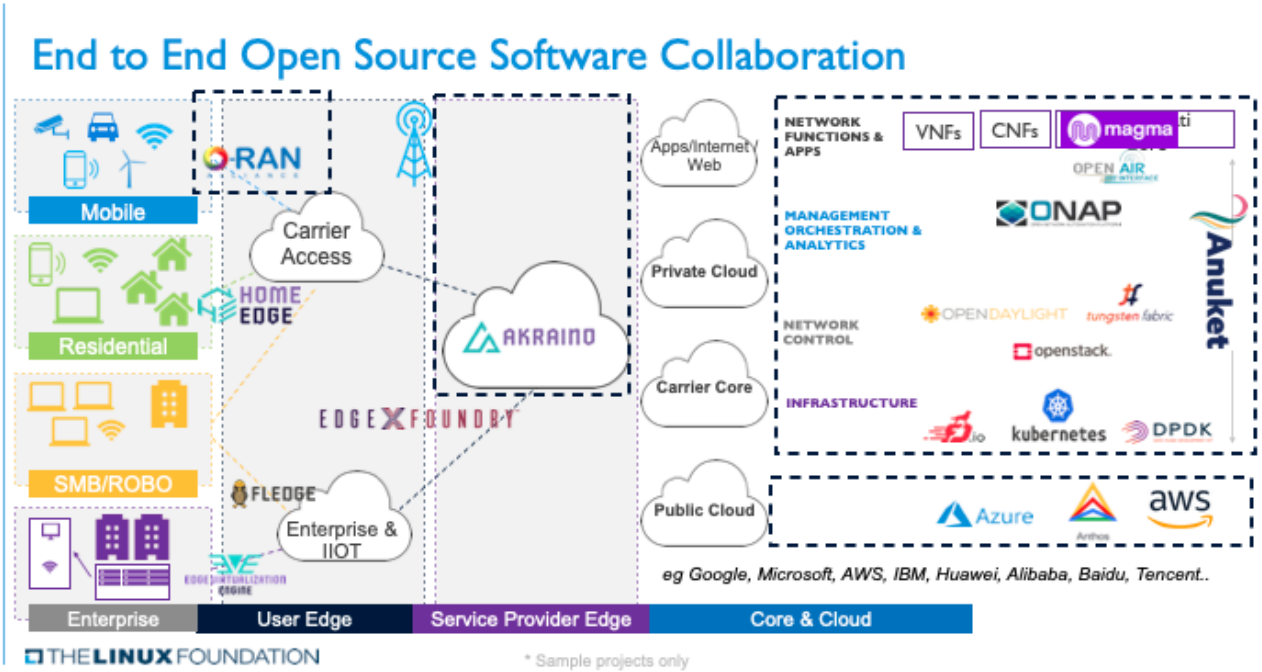
The Linux Foundation is also the home of nearly all the important open source projects in the 5G and networking space. Out of the $54B of the Linux Foundation community software projects that have been valued using the COCOMO2 model, the

open source projects assisting with building a 5G stack are estimated to be worth about $25B in shared technology investment. The LF Networking projects have been valued at $7.4B just by themselves.

The support programs at Linux Foundation provide the key foundations for a shared community innovations pool. These programs include IP structure and legal frameworks, an open and transparent development process, neutral governance, conformance, and DevOps infrastructure for end-to-end project lifecycle and code management. Therefore, it is uniquely suited to be the home for a community-driven effort to define an open source 5G end-to-end architecture, create and run the open source projects that embody that architecture, and support its integration for scaling-out and accelerating adoption.

The foundations of a complete open source 5G stack

The Linux Foundation worked in the telecommunications industry early on in its existence, starting with the Carrier Grade Linux initiatives to identify requirements and building features to enable the Linux kernel to address telco requirements. In 2013, The Linux Foundation’s open source networking platform started with bespoke projects such as OpenDaylight, the software-defined networking controller. OPNFV (now Anuket), the network function virtualization stack, was introduced in 2014-2015, followed by the first release of Tungsten Fabric, the automated software-defined networking stack. FD.io, the secure networking data plane, was announced in 2016, a sister project of the Data Plane Development Kit (DPDK) released into open source in 2010.



Linux Foundation & Other Open Source Component Projects for 5G At the time, the telecom/network and wireless carrier industry sought to commoditize and accelerate innovation across a specific piece of the stack as software-defined networking became part of their digital transformation. Since the introduction of these projects at LFN, the industry has seen heavy adoption and significant community contribution by the largest telecom carriers and service providers worldwide. This history is chronicled in detail in our whitepaper, Software-Defined Vertical Industries: Transformation Through Open Source.

The work that the member companies will focus on will require robust frameworks for ensuring changes to these projects are contributed back upstream into the source projects. Upstreaming, which is a key benefit to open source collaboration, allows the contributions specific to this 5G effort to roll back into their originating projects, thus improving the software for every end-user and effort that uses them.

The Linux Foundation networking stack continues to evolve and expand into additional projects due to an increased desire to

innovate and commoditize across key technology areas through shared investments among its members. In February of 2021, Facebook contributed the Magma project, which transcends platform infrastructure such as the others listed above. Instead, it is a network function application that is core to 5G network operations.

The E2E 5G Super Blueprint is being developed by the LFN Demo working group. This is an open collaboration and we encourage you to join us. Learn more here

Building through organic growth and cross-pollination of the open source networking and cloud community

Tier 2 operators, rural operators, and governments worldwide want to reap the benefits of economic innovation as well as potential cost-savings from 5G. How is this accomplished?

With this joint announcement and its DARPA supplier community collaboration, the Linux Foundation's existing projects can help serve the requirements of other large end-users. Open source communities are advancing and innovating some of the most important and exciting technologies of our time. It's always interesting to have an opportunity to apply the results of these communities to new use cases.

The Linux Foundation understands the critical dynamic of cross-pollination between community-driven open source projects needed to help make an ecosystem successful. Its proven governance model has demonstrated the ability to maintain and mature open source projects over time and make them all work together in one single, cohesive ecosystem.

As a broad set of contributors work on components of an open

source stack for 5G, there will be cross-community interactions. For example, that means that Project EVE, the cloud-native edge computing platform, will potentially be working with Project Zephyr, the scalable real-time operating system (RTOS) kernel, so that Eve can potentially orchestrate Zephyr devices. It’s all based on contributors’ self-interests and motivations to contribute functionality that enables these projects to work together. Similarly, ONAP, the network automation/orchestration platform, is tightly integrated with Akraino so that it has architectural deployment templates built around network edge clouds and multi-edge clouds.

An open source platform has implications not just for new business opportunities for government suppliers but also for other institutions. The projects within an open source platform have open interfaces that can be integrated and used with other software so that other large end-users like the World Bank, can have validated and tested architectural blueprints, with which can go ahead and deploy effective 5G solutions in the marketplace in many host countries, providing them a turnkey stack. This will enable them to encourage providers through competition or challenges native to their in-country commercial ecosystem to implement those networks.

This is a true solutions-oriented open source for 5G stack for enterprises, governments, and the world.

The post How open source communities are driving 5G’s future, even within a large government like the US appeared first on Linux Foundation.