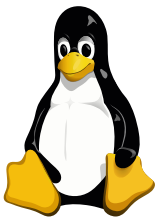
Linux kernel Bug Fixing Fall 2024

Abdul Rahim



December 5, 2024

Contents

Patches accepted

> Disclaimer: hyperlinks are italicized

```
$ git log --oneline \
    --author=abdul.rahim@myyahoo.com \
    --after="1 september 2024"

c152737be22b ceph: Use strscpy() instead of strcpy() in __get_snap_name()
95504d54a275 ACPI: thermal: Use strscpy() instead of strcpy()
8150408bfdb2 asus-laptop: prefer strscpy() over strcpy()
86ebc1fe902f usb: gadget: f_midi: prefer strscpy() over strcpy()
2a94a0898b14 selftest: alsa: check if user has alsa installed
e6ba83cb81e8 Documentation: PCI: fix typo in pci.rst
```

Please find all patches (in mainline and `linux-next`) here

In total, I had 7 patches accepted with 6 patches already mainlined. Please find bellow the summary of my works:

1. [PATCH] PCI: Fixed spelling in Documentation/PCI/pci.rst
    - *lkml link*
    - *commit link*

Status: mainlined

This was my first patch where I fixed a simple spelling mistake. I made a mistake by making 2 changes in a single patch that only describes one. I was politely corrected by Jonathan Corbet and Bjorn Helgaas.

2. [PATCH] usb: gadget: f_midi: prefer strscpy() over strcpy()
    - *lkml link*
    - *commit link*

Status: mainlined

In this patch I upgraded a depreciated and insecure API i.e. strcpy() which copies contents from a source buffer to a destination buffer, however the problem with this API is that it does not check the size of

the destination buffer, hence paving the way for linear overflow vulnerabilities. Imagine passing an argument where the source buffer is much larger than the destination buffer. In this case it would write beyond the destination buffer into kernel space memory potentially causing kernel panics. Here's the *documentation* for more information.

3. [PATCH] selftest: alsa: check if user has alsa installed

    - *lkml link*
    - *commit link*

Status: mainlined

So I was compiling `kselftest` when I noticed long warnings being printed. This was because I didn't have the `alsa` package installed. It is required to develop sound related software. I initially proposed a naive solution of adding a `--silent` flag which was obviously rejected. The problem with this solution was that it could silence other useful warnings, hence I changed my approach by adding a check to see if the user has `alsa` installed. Hence saving time by not attempting a compilation that is sure to fail.

4. [PATCH] asus-laptop: prefer strscpy() over strcpy() [commit]

    - *lkml link*
    - *commit link*

Status: mainlined

5. [PATCH v2] ACPI: thermal: Use strscpy() instead of strcpy()

    - *lkml link*
    - *commit link*

Status: mainlined

6. [PATCH v2] ceph: Use strscpy() instead of strcpy()

    - *lkml link*
    - *commit link*

Status: mainlined

7. [PATCH] fujitsu-laptop: replace strcpy -> strscpy

    - *lkml link*
    - *commit link*

Status: In `linux-next`

Patch 4-7 helps mitigate linear overflow vulnerabilities in the linux kernel. By replacing the strcpy() API with strscpy() API. According to the documentation, the API strcpy() is deprecated and it does not check the size of the destination buffer, hence making it insecure. The use of strscpy() is recommended. This is actually a common class of bugs that stems from how pointers work.

Other Patches

1. [PATCH] nfc: s3fwrn5: Prefer strscpy() over strcpy()
    - *lkml link*

The patch got a *Reviewed-by* tag from the maintainer. I need to resend it with a better changelog.

Summary of Experience

My time at the linux kernel mentorship program has been very rewarding. I learnt a lot in a very short span of time. I came with almost no background in kernel development. And now I am able to contribute to the kernel which I see as a big achievement. Initially when I applied for the program, I only had a little background in C and kernel, hence I started learning how the kernel is compiled, how the build process works, what are its subsystems, how patches are sent & applied and so on.

I was reading the Linux Kernel Mailing List (LKML) from August this year, but I understood very little of what was happening. People would paste a lot of code, and then someone would come and tell what was wrong with their code. My first impression was that these guys must be geniuses. Most of us would even struggle to understand the code and these guys are able to tell what is wrong with the code in just a glance. That to me was something quite extraordinary. Kernel development is widely regarded by some as a place where the epitome of programmers work. I wanted to contribute to the kernel because I wanted to prove to myself that I can contribute to large complicated projects like Linux Kernel.

From a background standpoint, In my second year at college I got hold of a book called “The C programming language” by Dennis Richie and Brian Kernigan. I really started liking the C programming language but the only place it is majorly used today is the kernel hence I knew I had to work in the Linux Kernel. While at LKMP, I started reading the “Linux device drivers” by Greg-kh, Jon and Allesandro Rubini. It gave me insights into kernel driver interface and other areas.

After reading the LKML for a while, I started to make sense of it. At around 2-3 weeks I sent my first patch in mid of september. It was a simple spelling fix, but it was important in that it introduced me to the process. How do you generate patches, how do you send it to developers and how to incorporate feedback and so on. Later on I worked on other patches and also came up with some ideas on how the Kernel can be improved.

In summary, I am glad I had the chance to participate in this program. I learned a lot of stuff and my technical skills improved so much. The experience I have gained here would not only be helpful in kernel development but to any field I go to. I made a few friends and got to learn so much from them. People at LKML are very nice, whenever I made a mistake, they politely corrected me. I would like to specially thank Shuah Khan who is running the program, I learned a lot from her. I really enjoyed my time here in the Linux Kernel, and I would love to contribute even more to the kernel. I plan to pursue kernel development as a hobby in the future.