Loading Intel Microcode

Introduction

Due to revelations the last years of hardware vulnerabilities with processors using speculative execution and various “threading” techniques, the Kernel has implemented a range of mitigations for these issues to eliminate the issue or reduce the potential problem. Alot of these solutions are included in the Kernel and can be/are activated in various ways.
However, some issues cannot be solved in the Kernel alone, which is why Intel and Amd released a bunch of Microcode updates to mitigate the issues. These Microcodes are normally included with BIOS updates, and have already been implemented on most computers. If you can update your bios, this is the best solution. Computer manufacturers/vendors normally release microcode updates with BIOS updates.
If everything else fails, you might have to download the microcode yourself and assure that it is loaded during boot of your Kernel/initrd, early in the boot process. This is not a permanent “update”, but rather a microcode loading facility in the Linux Kernel which is done at every boot.

The introduction is important, please read it carefully before proceeding. The best way to solve this and update microcode is by updating your BIOS/UEFI if possible.

Background

https://www.kernel.org/doc/html/latest/admin-guide/hw-vuln/index.html
This is a list of the known hardware vulnerabilities in the spectre/meltdown class of issues, with further information about how the Kernel handles it, or what can be done to solve it etc.

On your computer, your current status is listed during boot, but can also be found in the following way:

```
ls /sys/devices/system/cpu/vulnerabilities/
cat /sys/devices/system/cpu/vulnerabilities/*
```

The results greatly depend, but in most cases there are already active mitigations. Perhaps in your cases they are mostly mitigated and only SMT issues still show. If that is the case, you don't need a microcode update. If any of those entries say “no microcode”, you need to update your BIOS, OR load the relevant microcode on boot.
Kernel.org information about microcode loading:
https://www.kernel.org/doc/html/latest/x86/microcode.html

Although the method in this guide only describes **Intel microcode**, the method for AMD is very much the same. But the naming conventions are different, and there might be other minor differences. I can't test with AMD as I don't have it. If anyone can add AMD specific information in this article, that would be more than welcome.

Intel microcode information

This page contains Intels take:
https://www.intel.com/content/www/us/en/developer/articles/technical/software-security-guidance/best-practices/microcode-update-guidance.html
At the very end of this article, you can find a link to the official Intel microcode files.
https://github.com/intel/Intel-Linux-Processor-Microcode-Data-Files/tree/main/intel-ucode
Those files are ordered according to Intel CPU **“family”-“model”-stepping“** which might make it difficult to find your microcode. To find your CPU information, you can follow the Intel “readme” on the same page, OR you can do this:

```
dmesg | grep -i smp
```

which should return several lines including a line starting with “smpboot: CPU0” which contains the information (family: X, model: Y, stepping: Z). Read the information without the leading “0x” to get the exact ucode file you need to use. Example: 0x6,0x3a,0x09 is “06-3a-09”.
Download the correct ucode file and place it in the /lib/firmware/intel-ucode/ folder. In Slackware you have to create the “intel-ucode” folder in /lib/firmware/.

Building the microcode into the Kernel

For those in the habit of building their own Kernel, you can include the microcode in the Kernel and have the Kernel load it early in the boot. This should work independently of having an initrd or not (only tested without), but if you use an initrd you should load the microcode with the initrd instead. Loading it with initrd is the better method overall.
Only use this method if you don't use an initrd!
Including the microcode in the Kernel and having the Kernel load it at boot does NOT taint the Kernel. It is very simple to include the microcode in the Kernel. You build the Kernel as usual, but configure it with these two options: (referred to as “regular firmware method”)

```
CONFIG_EXTRA_FIRMWARE="intel-ucode/06-3a-09"
CONFIG_EXTRA_FIRMWARE_DIR="/lib/firmware"
```

Replace the ucode name in the example above with YOUR ucode name. Then build the Kernel. The options above can be found in menuconfig etc “Device Drivers” –> “Generic Driver Options” –> “Firmware Loader” –> “Firmware Loading Facility”. And the DIR option is shown once you put something in the first option.
This step only works if you already followed the instructions above and put the correct firmware in /lib/firmware/intel-ucode already

Once you have built the Kernel and updated your bootloader and booted into the Kernel, you can verify if the microcode was loaded:

```
dmesg | grep microcode
```

Which should give you information about the microcode. If it returns nothing, something went wrong with the loading, or it might return an error notice.

At this point it would make sense to check if any additional mitigations have been added. As before

```
cat /sys/devices/system/cpu/vulnerabilities/*
```

Loading Intel microcode with initrd

Not yet tested. Info will be added here. This is the preferred method, but can't be used if you don't use an initrd.

Meanwhile someone did this, but documented it elsewhere:
https://www.bernieland.com/share/howto-microcode_early_loading.html I will try to have the information added here asap.

AMD

Can't test with AMD, please add AMD info here.
The AMD microcode folder is /lib/firmware/amd-ucode/ and in Slackware this folder has to be created.

Servers - additional information

On the kernel.org page, there is a description for a method to load microcode after boot (no reboot). However, this microcode loading might not have any effect. Most likely it will load the microcode, but have no effect on the outcome. Having tested this method, the microcode did load, but mitigations were not effective.
This method could be used to test the loading of the microcode without rebooting.

Appendage

This article is not really completed.

In case you need some guidance with building your kernel:
brief_kernel_build
In case you need assistance with Grub legacy/bios mode:
lilo_to_grub_bios_mbr

Sources

* Original source: https://www.kernel.org/doc/html/latest/x86/microcode.html
* Original source: https://www.kernel.org/doc/html/latest/admin-guide/hw-vuln/index.html
* Original source: https://github.com/intel/Intel-Linux-Processor-Microcode-Data-Files
* Original source:

https://www.intel.com/content/www/us/en/developer/articles/technical/software-security-guidance/best-practices/microcode-update-guidance.html
* Original source:
https://github.com/intel/Intel-Linux-Processor-Microcode-Data-Files/tree/main/intel-ucode
* Written by zeebra

howtos,, security,, cpu,, microcode,, spectre,, meltdown