Episode Book

EPISODE #6353

How Ubuntu Actually Updates Your Firmware

PUBLISHED	29 October 2026 at 18:57 UTC
DURATION	30:13
LISTEN	myweirdprompts.com/episode/ubuntu-uefi-capsule-fwupd-47482/
HOSTS	Corn, Herman, Bruno, Hilbert
SCRIPT MODEL	openai/gpt-6.1-sol
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.25
SOURCES	13 cited by the research (listed at the end)
BOOK GENERATED	30 October 2026 at 07:19 UTC

SYNOPSIS

Operating systems routinely prompt users to install hardware firmware, but the underlying mechanisms differ drastically from regular software patches. A single desktop notification can mask completely different operations, from early-boot CPU microcode patches to live peripheral flashes and UEFI capsule updates. This look at fwupd and the Linux Vendor Firmware Service details how Linux negotiates device updates, traverses Secure Boot with shim, and relies on preboot firmware to perform the final write to motherboard flash.

TRANSCRIPT



Corn

The reassuring pitch is that Ubuntu makes firmware updates just another software update. The useful part is that they are not. Daniel sees those hardware scans alongside his regular upgrades and wants to know what actually happens after he agrees. His image is an iron wall between the operating system and the BIOS. How does something downloaded by Ubuntu get through that wall, and why does CPU microcode seem to belong in the same conversation?



Herman

The notification compresses several different transactions into one friendly rectangle. Replacing a browser library, staging motherboard firmware, and installing a CPU microcode package can all begin on the desktop. They do not end in the same place.



Corn

The rectangle has omitted some material differences.



Herman

Especially who performs the final write. With the usual UEFI capsule route, Ubuntu prepares an update, then hands execution over so the machine's firmware can apply it. With CPU microcode, the Linux kernel submits a processor-specific patch directly to the processor, normally very early in boot. Different interfaces, different persistence, different recovery problems.



Corn

That last distinction matters to me. Rebooting after a package upgrade usually means discovering which application has acquired a new opinion about my settings. Rebooting after motherboard firmware means trusting code that has to work before Ubuntu can complain about it.



Herman

An ordinary apt transaction mostly replaces files in storage that Linux manages. There can be services restarting and installation scripts running, so it is more than copying files, but the operating system understands the destination. Firmware installation involves a



Herman (continued)

hardware-specific update interface and a device that decides what it accepts.



Corn

I also want to separate downloading from installing. A package can sit on disk harmlessly while the interface makes it look as though the machine has already been improved. With microcode, the package being current and the processor running the new revision are different facts.



Herman

Hold that distinction. It becomes very important. On the device side, the central service is fwupd, developed by GNOME maintainer Richard Hughes and first released on March seventeenth, two thousand fifteen. It talks to the Linux Vendor Firmware Service, LVFS. Vendors upload firmware and matching information there. Ubuntu's Firmware Updater is a front end to that machinery.



Corn

Before you fall in love with another daemon, Herman, what problem did the daemon actually remove?



Herman

Hunting through a manufacturer's website for a Windows executable that Linux cannot run. That was a real obstacle, not merely an ugly interface. Users often did not know the exact installed component, its firmware revision, or whether an update existed. LVFS describes systems staying insecure even when updates have already been released that fix their specific problems.



Corn

The patch existed. The compatible scavenger hunt did not.



Herman

Hardware discovery, matching, delivery, and authorization become one supported workflow. Ubuntu has included its separate Flutter-based Firmware Updater since Ubuntu twenty-three ten. The command-line client, fwupdmgr, exposes the same broad operations, refreshing metadata, enumerating devices, finding updates, and requesting installation.



Corn

Meanwhile, the processor packages still come through apt. That is why I would not infer the mechanism from where the notification appeared. An interface can collect messages without becoming the thing that performs every update.



Herman

Would you want those messages separated completely, Corn, or would that just recreate the problem of firmware being forgotten?



Corn

Together for discovery, separated for consent. Tell me whether I am replacing a file, changing persistent device code, or preparing a patch for the next boot. I do not need three applications. I need the one application to stop pretending those are equivalent errands.



Herman

Then the next useful detail is ESRT, the EFI System Resource Table. The system firmware exposes a list of updatable resources to the operating system. Linux can discover that the machine supports a particular capsule update, along with identifying and status information. Linux support for this route arrived with kernel four point two.



Corn

The firmware advertises its own update entrance. Ubuntu is not finding an unguarded basement window.



Herman

Right. Well... yes, for this standardized route. Older flash utilities and other hardware can use different mechanisms, so I would not make a universal claim that an operating system can never write flash. Here, fwupd is coordinating a firmware-controlled installation. The update interface is deliberately available to privileged software.



Corn

That makes the iron wall less mysterious. Isolation does not require there to be no interface. It requires the interface to accept only the right operations, from the right caller, with the right payload. The uncomfortable part is how much work those three qualifications contain.



Herman

Start with the payload. fwupd identifies supported devices through plugins. Those plugins assign GUIDs, and the update metadata specifies which GUIDs a release matches. It does not simply compare the friendly name displayed in the window.



Corn

Two laptops called the same thing can contain different boards. Marketing names have never struck me as adequate flash-programming instructions.



Herman

A plugin can incorporate identifiers that distinguish the actual component or platform. The daemon consults LVFS metadata in AppStream format, with a detached GPG or PKCS signature. The downloaded cabinet archive carries component metadata and payload authentication information. Those checks establish provenance and integrity before installation proceeds.



Corn

Provenance still does not mean compatibility by itself. A perfectly genuine image for the wrong board remains the wrong image. I would rather hear those as separate checks than hear the word signed doing every job in the sentence.



Herman

They are separate. Matching identifies the intended target. Distribution signatures authenticate material delivered through the service. The firmware may also validate a vendor-signed capsule at its own boundary. Authorization on the host is another layer again, with PolicyKit negotiating whether the requesting user may perform the operation.



Corn

Several locks, several different keys. That is less elegant than one magic signature, and considerably more believable.



Herman

Once the user agrees to a supported UEFI update, fwupd unpacks the firmware payload onto the EFI System Partition. That partition is storage accessible to the preboot environment. It sets the UEFI variables needed to arrange the update path, and the system reboots.



Corn

Notice that nothing has yet required Ubuntu to overwrite its own BIOS. It has prepared files and instructions for the next stage. This is also where a message saying downloaded or scheduled can be technically accurate while telling me very little about whether installation has happened.



Herman

On reboot, the fwupd EFI executable runs before the normal bootloader. It invokes the firmware's UpdateCapsule service. The firmware receives the capsule through that interface and performs the platform-specific processing and installation. That is the handoff Daniel is asking about. Linux prepared the request. Firmware controls the persistent change.



Corn

The operating system gets to book the appointment.



Herman

It also has to deliver the correct patient.



Corn

Your retired pediatrician voice has entered the firmware discussion.



Herman

Some habits survive retirement. In this case, the identification really does matter. A successful download is only one stage. Scheduling the reboot is another. Firmware accepting and applying the capsule is another. Afterward, the system has to report the result back.



Corn

Secure Boot seems like the obvious objection. We have spent considerable effort deciding what may execute before Linux. Then an updater drops another executable into that interval and says it has business there.



Herman

fwupd uses shim to chainload its EFI updater on Secure Boot systems. The updater participates in the trusted execution chain rather than requiring Secure Boot to be switched off. There are two separate authenticity questions here… no, let me narrow that. At least two. May this EFI program execute, and may this firmware payload be installed? Answering the first does not automatically answer the second.



Corn

That explains why disabling Secure Boot should not be the assumed price of receiving firmware fixes. The update route was designed to coexist with it. I would be suspicious of a security workflow whose opening move was removing an unrelated protection.



Herman

Practical installation details can still make it feel fragile. The machine needs to be booted in UEFI mode for this capsule path, not legacy BIOS mode. The EFI System Partition has to be available. Some documented arrangements depend on udisks being present, and even the expected uppercase EFI directory name can matter.



Corn

Civilization survives another capitalization requirement.



Herman

Those are implementation details, not the fundamental update protocol, but they explain some apparently irrational failures. A machine can support firmware updates while its current Linux installation does not provide the staging environment the updater expects.



Corn

I can see your diagnostic expression, Herman. You are about to describe a missing partition mount as encouraging.



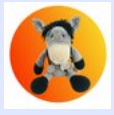
Herman

Compared with corrupt flash, it is encouraging.



Corn

I concede the diagnosis. I retain the complaint.



Herman

There is also a post-update boot issue that can be mistaken for a ruined motherboard. A firmware update can discard boot configuration, including the NVRAM entry that points to the Linux bootloader. Recreating that entry with efibootmgr may restore booting even though the firmware update itself succeeded.



Corn

That difference changes the whole conversation with the owner. The machine may have forgotten how to reach the installed system rather than destroyed it. Both present as no Ubuntu, but only one justifies the much more expensive expression on your face.



Herman

What would worry you most during the installation itself, a progress bar that stalls or a completely blank screen?



Corn

The blank screen. A stalled bar at least supplies a suspect. Blankness invites every possible theory, including the theory that pressing the power button will help.



Herman

That is why model-specific behavior matters. The Lenovo ThinkPad P fifty is a documented example where the flash can proceed with a black screen and no messages. Interrupting or force-resetting during that operation can turn an uninformative display into a damaged installation. Silence is not evidence that the write has stopped.



Corn

An interface should not require faith in an invisible progress bar.



Herman

Better progress reporting would remove a lot of avoidable anxiety. Still, the same fwupd notification may lead to a much less dramatic operation on another device. Many USB, Thunderbolt, and touchpad updates can run while Linux stays up. The plugin speaks the device's update protocol, transfers the image, and the device performs its own installation.



Corn

Live means the whole computer need not reboot. It does not promise the particular device remains useful throughout. Losing a dock briefly could remove the screen displaying your reassuring updater.



Herman

Storage firmware especially needs that distinction. Some SSD update paths can transfer an image live but need a reset or power cycle for activation. Requirements vary by device. The daemon collects those requirements and decides whether to install immediately or stage work for a later boot.



Corn

My touchscreen already regards these claws as unauthorized peripherals. I would appreciate advance notice before the touch controller disappears as well.



Herman

The point of the plugin architecture is precisely that a touch controller, an SSD, and motherboard firmware do not all obey one protocol. The common service provides discovery, policy, matching, and status around different device-specific operations.



Corn

CPU microcode breaks even that pattern, though. There is no reason to send its package through the motherboard capsule route if the processor already exposes a patch-loading interface to the kernel.



Herman

Ubuntu normally delivers it through the intel-microcode or amd64-microcode packages. Installing an updated package refreshes the early boot material in the initramfs. That is the initial filesystem image the kernel receives at boot, and it can begin with an uncompressed cpio archive containing the processor patches.



Corn

The package installation changes tomorrow's boot ingredients. It has not necessarily changed today's running processor. That also explains why a machine can appear fully upgraded and still need its reboot for the security change to take effect.



Herman

The early loader scans that material before normal multiprocessor startup. It checks for a patch matching the CPU, applies it on the bootstrapping processor, then handles the other processors as they come online. The kernel caches the patch so it can reapply it when needed after sleep.



Corn

Firmware can have loaded an earlier revision first, can't it? Otherwise an old motherboard would leave the processor completely dependent on Linux arriving with a correction.



Herman

Firmware commonly loads microcode during initialization. The kernel's early loader can then supply a newer suitable revision, which is valuable when the motherboard vendor has stopped releasing firmware. An operating system update can close a processor vulnerability without waiting for a new motherboard image.



Corn

That is a substantial extension of hardware life. The board manufacturer no longer has the only practical route to correcting the processor. Though it is still the processor vendor supplying the patch, rather than Ubuntu rewriting Intel's internal instructions over breakfast.



Herman

The kernel delivers a vendor-produced blob through a privileged hardware interface. On Intel, the update trigger involves a write to a model-specific register using the wrmsr instruction. The CPU checks the patch's applicability and accepts it through its update mechanism. AMD has its own processor-specific details, so that Intel register sequence should not be presented as universal.



Corn

Root does not become a microcode author merely by finding the register.



Herman

Correct. The privilege allows initiation of the operation. The processor still governs acceptance. This is a different kind of boundary from the motherboard flash procedure. The kernel is already executing and directly asks the processor to activate a matching patch.



Corn

What physically changes inside it? My first guess would have been some tiny rewritable memory replacing the original control program, but that sounds rather permanent for something reapplied after boot.



Herman

The original microcode ROM stays in place. In the classic Intel description, the patch occupies separate SRAM, with match registers redirecting selected internal sequences from ROM into the patch. You are correcting particular behavior, not replacing every bit of the processor's internal implementation.



Corn

The silicon receives an errata sheet it can execute.



Herman

That capability became important for field corrections with the P six generation. The Pentium floating-point division disaster helped establish why manufacturers wanted a way to address certain processor bugs without recalling the physical product. That does not mean every hardware defect can be fixed with microcode. The available hooks and the nature of the defect constrain what is possible.



Corn

Nor was the original Pentium division fault repaired by everyone downloading a convenient package. The historical motivation and the later mechanism should stay separate. Otherwise we accidentally give a very expensive recall a happy software ending it never had.



Herman

Yes, that is the distinction. Microcode is powerful within the parts it can influence. It cannot invent missing physical resources or repair damaged transistors. Some fixes alter instruction handling, internal checks, or the controls the operating system can use for mitigation.



Corn

Herman, you like updates that avoid downtime. I can practically hear you wondering whether all this could happen without restarting.



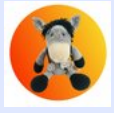
Herman

I was about to explain why that is a bad default.



Corn

Personal growth at the model-specific register.



Herman

Earned through documentation, in this instance. Since kernel five point nineteen, late microcode loading has not been enabled by default. The kernel documentation has a section actually titled Why is late loading dangerous? That is unusually direct documentation.



Corn

Hm.



Herman

Simultaneous multithreading siblings can share a microcode engine. During an update, processor behavior that running software relies on can change underneath it. Some model-specific registers provided by microcode can transiently disappear. Certain exceptional events, including machine checks and nonmaskable interrupts, cannot simply be excluded from the whole operation.



Corn

Stopping normal application work therefore does not create a perfectly quiet processor. There are still things that can arrive while the change is underway. Early boot reduces how much software has already committed itself to the old behavior.



Herman

Early loading also ensures the kernel initializes against the corrected processor behavior from the outset. For severe errata affecting families such as Haswell and Broadwell, that early timing can be essential. A patch applied after initialization may arrive after the dangerous decision has already been made.



Corn

This is one place where asking me to reboot is more convincing than promising heroic cleverness. I would rather interrupt my session than discover that avoiding the interruption required coordinating every exceptional event inside the CPU.



Herman

Would you apply that preference to servers too, Corn? A reboot can interrupt something more substantial than your browser.



Corn

I would distinguish the cost from the mechanism. Servers make the maintenance window more expensive, not the late-loading problem less real. Arrange redundancy or a window if the system needs one. Do not turn service availability into evidence that a risky processor transition is safe.



Herman

Zenbleed gives us a concrete reason this pipeline matters. The vulnerability affected AMD Zen two processors and could expose data across execution boundaries. Ubuntu distributed updated AMD microcode at coordinated disclosure time through its security packaging. The package route allowed the correction to reach affected machines without waiting for every motherboard vendor's BIOS release.



Corn

That is a much stronger argument than the vague promise of better stability. A processor bug can undermine boundaries that perfectly ordinary software assumes it has. The patch changes behavior below those assumptions.



Herman

Performance headlines can be misleading too. A corrected microcode revision may let the kernel use a more targeted mitigation than a broader software workaround. Sometimes a security fix costs performance, but comparing patched against unpatched without considering the operating system's alternative mitigations gives an incomplete answer.



Corn

The update can be slower than doing nothing and faster than doing the equivalent protection elsewhere. Those comparisons are different, and a benchmark title can conceal which one it used.



Herman

We should pin down the terminology before the security infrastructure part. Bruno is joining us as honorary chair of the Institute for Applied Capsule Gastronomy, author of The Motherboard Must Rest. Bruno, you are our expert today. When Ubuntu patches the firmware and the microcode, what are we actually doing?

B

Bruno

Firmware. In my family we call this the curing of the prosciutto. You do not cook it, you do not argue with it, you hang it in the cellar and you wait, and the cellar does the work, not you. The operating system is only the boy who carries the salt.



Herman

Specifically, on the UEFI side, what crosses from the operating system into the update interface?

B

Bruno

The capsule, this is a sealed parcel, like a sausage in its skin, signed and stamped. The machine reboots and the firmware opens the parcel before the operating system has even woken up. Nobody reaches through the iron wall. The wall has a letterbox and the firmware reads its own post.



Herman

Could you distinguish the processor's volatile patch from that persistent installation?

B

Bruno

Microcode, though, that is not a parcel at all. That is a whisper into the ear of the processor at the very first moment of boot, from the inside. My mother had a recipe like this, you cannot write it down, you must be there.



Herman

The operating system can carry it, though. What makes the correction important?

B

Bruno

Why does it matter for security? Because a bad cut spoils the whole dinner, my friend. The firmware runs before everything. If it is rotten, no sauce, no software, no cleverness on top will save the plate.



Herman

Can we separate that from the processor case?

B

Bruno

And Corn. Corn. You eat only leaves. This wounds me, truly, like a bad review of my opera. You and your brother come to my Sunday lunch, both of you, and I will make you a saltimbocca that sings. It is not negotiable, eh.



Corn

The updater's own security is the less comfortable half. We add privileged parsers, metadata handling, network fetching, and device protocols so we can remove vulnerabilities elsewhere. Each additional component needs maintaining too.



Herman

Passim is a concrete example. Integration introduced around fwupd one point nine point five added a local caching service intended to reduce repeated downloads. It listened on port twenty-seven thousand five hundred, and vulnerabilities were reported within weeks. Exact exposure depends on configuration and network access, but a cache daemon is still additional reachable software.



Corn

The patch delivery system has acquired a patch delivery requirement.



Herman

It does not erase the value of fwupd. It means the service has to be evaluated as software with privileges and inputs, rather than treated as automatically safe because its purpose is security. Download authentication cannot protect against every bug in the code parsing the download.



Corn

Automatic also needs unpacking. Downloading metadata in the background is not flashing a motherboard in the background. A daily notification is not an installation. An unattended apt transaction delivering a microcode package is not the same authorization event as agreeing to a firmware flash.



Herman

LVFS explicitly describes user agreement before the firmware update action. Desktop presentation and managed deployments can differ, but discovery, downloading, authorization, installation, and activation are separate stages. Calling all of them automatic obscures which one actually happened.



Corn

Ubuntu Pro and Livepatch belong elsewhere in this picture too. A kernel livepatch changes selected running kernel code. It is not the processor's microcode loader, and it is not a UEFI capsule installer. Similar promises about fewer unpatched machines do not make the mechanisms interchangeable.



Herman

The consequence of failure differs as well. An ordinary package update can break booting, and installation scripts can do serious damage, so software upgrades are not harmless. Persistent firmware corruption can remove the environment you would normally use to recover. Available recovery facilities then depend heavily on the device.



Corn

I would not want consent reduced to acknowledging a redder warning box. Tell me the expected interruption and what counts as success. If the updater says complete after transferring the file but before activation, it has left the most interesting verb unstated.



Herman

Version reporting deserves that same precision. A displayed value may come from device data, cached information, or a particular plugin's interpretation. Incidentally, that emphasis on discoverability was present from Richard Hughes's original work as a KDE maintainer, making the hardware state legible rather than requiring a vendor utility.



Hilbert

GNOME maintainer. You had it right earlier. My neighbour Martin had a desktop board that put a different revision in its setup screen from the one the Windows utility displayed. This was in two thousand three. I wanted that board myself. It was a hundred and eighty-nine pounds, and I left it in the shop. Martin bought his and asked me round when the update finished.



Corn

Two version strings would have made the evening longer.



Hilbert

They did. One said the old revision. One said the new one. Herman, you separated downloaded from installed earlier. Martin's utility did neither. It said successful in a green box and kept the old number underneath. He had made tea before starting. He put mine away from the keyboard because an anteater's snout leaves very little room between cup and screen.



Herman

A stale utility display could survive a successful flash. It would still need checking against what the board reported after reboot.



Hilbert

Take that small knock off my microphone out. We rebooted. Setup showed the new revision. The utility still showed the old one. I wrote the setup revision on the back of his envelope. Then I copied the utility's number beside it and underlined that one in red. I dislike green success boxes. They encourage people to stop reading.



Corn

Red ink does make an unresolved number harder to ignore.



Hilbert

Martin wanted to reconnect the network cable. I wanted another restart with the utility closed. He waited. He had already cut the cake into three pieces, although there were only two of us, because he knew I would ask for another. The utility finally showed the new revision after he shut it down completely. We had spent forty minutes checking an update that took four minutes.



Herman

Forty minutes of uncertainty is a cost the update interface ought to reduce. As more of this becomes automatic, can it preserve meaningful agreement about persistent changes and give a clear account of what actually activated? Ubuntu is staging a capsule for firmware-controlled installation in one case, and preparing a kernel-loaded processor patch in the other. The status needs to reflect that difference.



Corn

Thanks to Hilbert Flumingtop for producing the show.



Herman

If you enjoyed this one, try episode thirteen seventeen, The First Second; episode twenty-five ninety-four, The Hierarchy of Immutable Code; and episode fifty-nine sixty-seven, How UEFI, Secure Boot, and the Boot Chain Actually Work. This has been My Weird Prompts, the human-AI collaboration podcast. No question too obscure, no rabbit hole too deep.



Corn

If you enjoy the show, leave us a review wherever you listen. It helps other curious people find us.



Herman

I'm Herman Poppleberry.



Corn

I'm Corn. See you tomorrow.

SOURCES

What the research for this episode read before the script was written. Primary sources first.

1. **LVFS documentation, Introduction [primary]**
   https://www.fwupd.org/lvfs/docs/introduction

2. **The Linux Microcode Loader, kernel.org [primary]**
   https://www.kernel.org/doc/html/latest/arch/x86/microcode.html

3. **Wikipedia, last edited 2026-10-21**
   https://en.wikipedia.org/wiki/Fwupd

4. **Wikipedia, last edited 2026-06-05**
   https://en.wikipedia.org/wiki/Intel_microcode

5. **Wikipedia, last edited 2026-10-27**
   https://en.wikipedia.org/wiki/Microcode

6. **ArchWiki**
   https://wiki.archlinux.org/title/Microcode

7. **ArchWiki**
   https://wiki.archlinux.org/title/Fwupd

8. **A look at Ubuntu Desktop LTS 24.04, LWN**
   https://lwn.net/Articles/971143/

9. **Ars Technica, April 2024**
   https://arstechnica.com/gadgets/2024/04/ubuntu-24-04-lts-noble-numbat-overhauls-its-installation-and-app-experience/

10. **Ask Ubuntu (Firmware Updater, Ubuntu 23.10+)**
    https://askubuntu.com/questions/1238999/is-there-a-gui-for-the-tool-fwupd-firmware-updater-in-ubuntu-gnome

11. **grub-devel mailing list, August 2023 (Ubuntu early initrd microcode)**
    https://lists.libreplanet.org/archive/html/grub-devel/2023-08/msg00010.html

12. **Linux.org forum thread on firmware-updater on 24.04**
    https://www.linux.org/threads/firmware-updater-on-ubuntu-24-04-for-workstations.49821/

13. **Proxmox VE docs, CPU microcode section**
    https://github.com/proxmox/pve-docs/blob/3da833f5/firmware-updates.adoc