Episode Book

EPISODE #5967

How UEFI, Secure Boot, and the Boot Chain Actually Work

PUBLISHED	14 October 2026 at 15:05 UTC
DURATION	32:36
LISTEN	myweirdprompts.com/episode/uefi-secure-boot-chain-explained-17416/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	deepseek-flash
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.4
SOURCES	15 cited by the research (listed at the end)
BOOK GENERATED	19 October 2026 at 17:42 UTC

SYNOPSIS

If you've run Linux on the desktop for years but never got a straight explanation of firmware, this one's for you. We trace the arc from legacy BIOS boot — where the firmware executes 440 bytes of MBR code and then forgets you exist — through UEFI's structural shift to filesystem-aware firmware launching .efi applications off the EFI System Partition. Then we cleanly separate UEFI from Secure Boot, walk the PK/KEK/DB/DBX key hierarchy, and explain why Ubuntu's chain validates the shim, GRUB, and kernel but leaves the initramfs unchecked. Along the way: why the CSM was never part of the UEFI spec, why the early UEFI bug reputation was earned, and what the Microsoft third-party UEFI CA expiration in 2026 means for machines that never got a firmware update. Practical, concrete, and finally explained.

TRANSCRIPT



Corn

Twenty years of desktop Linux and there is still one part of the stack that nobody ever sat me down and explained properly. People will explain the kernel to you for six hours. Nobody explains the firmware.



Herman

It's the part you only meet when something has already gone wrong.



Corn

Daniel's been running Linux on the desktop for about twenty years, and he says the relationship between legacy BIOS booting, UEFI, and Secure Boot is the one thing he never felt he properly understood. And he's asking the question underneath it, which is a good one. How much of this is still live in twenty twenty-six, and how much is baggage we're all carrying because we learned it once and never got told it expired.



Herman

Which is honestly the right way to frame it. Because the concepts people carry around are not wrong, they're just old. The BIOS, the MBR, the boot order menu, the CSM setting you toggle when an install fails. Some of that is dead, some of it is still doing work, and one piece of it was never what people think it was in the first place.



Corn

And the piece he wants cleanly separated, because they get mashed together constantly, is UEFI versus Secure Boot. He wants the history, he wants the anatomy of the boot chain itself, he wants the newer stuff, UKIs and measured boot, and then he wants to land on the practical question. If you install a current Ubuntu LTS on modern hardware, what of this do you actually need to understand.



Herman

Let's do the arc properly. How we got here, then the distinction, then the chain, then where it's heading.



Corn

Start with how we got here.



Herman

Legacy BIOS boot is worth understanding precisely because of how crude it is. The firmware wakes up, and it executes the first four hundred and forty bytes of the master boot record on the first disk. That's the whole contract. Four hundred and forty bytes of code, no filesystem awareness, no real partition table awareness beyond whatever the boot code implements itself. Then that tiny thing chain-loads a second stage from somewhere else, the gap after the MBR, a volume boot record, or on a GPT disk a dedicated GRUB BIOS boot partition.



Corn

Four hundred and forty bytes.



Herman

That's your operating system's first breath. And past that point the firmware is basically done. It hands over and forgets about you.



Corn

A firmware that shows up, does one thing, and leaves. There's something honest about that.



Herman

There's something limited about it, which is why it got replaced. UEFI does something structurally different. It reads boot entries out of NVRAM, and it launches EFI applications, actual .efi files, stored on the EFI System Partition. It does not execute MBR boot code at all. And the single capability that unlocks everything downstream is that it can read a filesystem natively. The spec mandates FAT12, FAT16 and FAT32 support, so the firmware can open a partition and find a file by name.



Corn

Which is the whole difference.



Herman

The whole difference. Once firmware can read a filesystem, your bootloader stops being a magic block of sectors in a fixed location and becomes a file. A file you can list, replace, sign, hash, copy to another partition. Everything people find confusing about modern boot comes out of that one change.



Corn

And multiboot stops being a chain-loading problem.



Herman

Right. Under BIOS, two operating systems meant one bootloader had to load another bootloader, and they had to agree about it. Under UEFI each OS keeps its own directory on the ESP, its own vendor folder, its own .efi file, and the firmware just launches a different application. That's why dual-boot got structurally simpler even though it still feels fragile in practice, because the fragility moved into the firmware's boot order and the vendor folders rather than the chain-loading.



Corn

So where does the CSM fit. Because that's the thing I remember being told to toggle.



Herman

The Compatibility Support Module is UEFI's legacy-boot shim. If you enable it, UEFI generates CSM boot entries for all the drives, and picking one of those makes the CSM boot from that drive's MBR bootstrap code, exactly the old way. So it's a compatibility path, and it let people install on hardware whose boot media or whose OS didn't speak UEFI yet.



Corn

And here's the correction I want on the record, because I carried the wrong version of this for years. The CSM was never part of the UEFI specification.



Herman

Never. It was specified by Intel in its Platform Innovation Framework for EFI. It was always a transitional layer, bolted on from outside the spec, meant to be removed once nothing needed it.



Corn

Which means the whole "UEFI is just BIOS with a CSM bolted on" mental model is backwards. The legacy part was the bolted-on part.



Herman

And it's now largely gone. Intel's own guidance says all future products will only be validated as UEFI Class 3 platforms, which means no CSM at all. They pushed motherboard partners to remove it by twenty twenty. VMware deprecated legacy BIOS and CSM in vSphere seven and later. So when Daniel asks whether legacy BIOS boot is effectively obsolete on contemporary desktop hardware, the honest answer is yes. It survives on older machines and in virtual machines. On a new desktop it's not a choice you're making anymore.



Corn

What about the distrust, though. Because a lot of people my age learned to hate UEFI for a reason, and it wasn't ideology.



Herman

It was bugs. Early vendor UEFI implementations, call it pre-twenty-thirteen, could carry more bugs than the BIOS they were replacing. They'd refuse to boot a perfectly valid install, they'd lose the boot entry on a firmware update, they'd have a UI designed by somebody who had never installed an operating system. That's where the muscle memory comes from. And it's worth naming, because it explains the reflex without justifying it today.



Corn

The firmware grew up. The reflex didn't get the memo.



Herman

The hardware got the memo. The people didn't, which is fair, because nobody sends you a letter.



Corn

So that's raw sectors to EFI applications. Now the part that actually confuses everyone, which is that UEFI and Secure Boot get used as if they're the same word.



Herman

They're not, and the separation is clean once you see it. UEFI is the firmware interface. Secure Boot is an optional feature of the UEFI specification. You can have UEFI with Secure Boot off, and plenty of machines ship that way. When it's enabled, the firmware does one specific thing. It refuses to launch any EFI executable that isn't signed by a key in its trust databases.



Corn

That's it.



Herman

That's the entire mechanism. It's a signature check at launch time. Everything complicated about Secure Boot is downstream of that one sentence, and everything painful about it is key management.



Corn

Walk the keys, because that's where the teeth are.



Herman

The hierarchy runs Platform Key, PK, owned by the OEM, then the Key Exchange Key, KEK, which authorises updates to the databases, then DB, which holds the allowed signatures, and DBX, which holds revoked signatures and hashes. If you want to revoke a bad signed binary, you add its hash to DBX. That's the whole revocation system, and it's why Microsoft can push a revocation out to the entire fleet.



Corn

And the failure state. Because there is one, and it's ugly.



Herman

A system with no PK enrolled is in Setup Mode. In Setup Mode, any authenticated variable can be written without signature verification. So if you leave a machine in Setup Mode with Secure Boot nominally on, that's a complete bypass. The check is still there on paper and the key store will accept anything. It's the security equivalent of a lock with the door taken off.



Corn

A deadbolt on a doorway with no wall.



Herman

That's the image, yes. And the reason Linux users run into Microsoft here is structural rather than sinister. On most x86 hardware the DB contains Microsoft's certificates, so a shim signed by Microsoft is what the firmware will accept. Every mainstream distribution ships a shim, and every one of those shims is signed by Microsoft. That's a real supply-chain dependency and it's honest to say so out loud rather than pretend it isn't.



Corn

And Ubuntu's chain specifically.



Herman

Firmware validates the Microsoft-signed shim. Shim validates a Canonical-signed GRUB, or MokManager. GRUB loads and validates the Canonical-signed kernel. Ubuntu has supported this since twelve oh four LTS, and since eighteen oh four it verifies the bootloader, the kernel and the kernel modules. And then there's the hole, which Ubuntu's own documentation states plainly. The initrd images are not validated by that chain.



Corn

Say that again, because it's the part I keep thinking about.



Herman

The initramfs, the early userspace archive the kernel loads before it can mount your real root filesystem, is not signed and not checked in that chain. An attacker with write access to the boot directory can swap the initramfs entirely and the chain will never notice.



Corn

So Secure Boot verifies the bootloader and the kernel and then politely looks away at the exact moment the system starts running code that mounts your encrypted disk.



Herman

It closes that gap later, and we'll get to how. For now, that's the state of the default Ubuntu chain.



Corn

Custom kernels. This is the practical friction Daniel asked about and it's the bit that eats an afternoon.



Herman

Any module not built into the kernel has to be signed. Canonical-signed modules load without you doing anything. A module you compiled yourself needs a Machine Owner Key, a MOK. Unsigned modules just fail to load with an error, and the error doesn't always tell



Herman (continued)

you it's a signing problem. So you enroll a key with mokutil import, and at the next boot MokManager presents it for physical confirmation with a one time password.



Corn

Physical confirmation. Somebody has to be standing at the machine.



Herman

That step is the design, not an annoyance. If enrollment could be completed entirely from a root shell, a remote attacker who got root once could enroll a key and persist across reboots through Secure Boot itself. Requiring a human at the keyboard at the next boot breaks that. It's the one place in this whole story where a physical presence check actually earns its keep.



Corn

And the failure people actually hit.



Herman

Ubuntu twenty six oh four LTS users ran into Secure Boot blocking unsigned DKMS module loads when the MOK enrollment or the signing was missing. Which is the abstract key management problem showing up as a real error message on a real desktop. You install a third-party driver, it works on the machine you built it on two years ago, and on the new install it silently doesn't.



Corn

Because the key store isn't the same key store.



Herman

Because the key store isn't a part of the distro. It lives in firmware.



Corn

So that's the mechanism and the friction. Now the current story, because there's one happening right now and it's not background.



Herman

The Microsoft third party UEFI CA, the certificate that signed every Linux shim for over a decade, expires in twenty twenty-six. Microsoft is splitting the responsibilities. Windows UEFI CA twenty twenty-three covers bootloaders and shims, Microsoft Option ROM UEFI CA twenty twenty-three covers hardware firmware, and there's a new KEK, Microsoft Corporation KEK two K CA twenty twenty-three.



Corn

And this is not a clean swap, because firmware is involved.



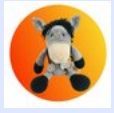
Herman

Firmware is the whole problem. New OEM hardware ships with only the new CAs pre-enrolled. Older hardware has the old CA and gets the new one through firmware updates, which for a lot of machines never arrive. And the failure mode is specific. On a system where the old CA is removed before the new one is enrolled, a shim signed only by the old CA fails to boot. Not warns. Fails.



Corn

So Rocky ships dual signed shims.



Herman

Rocky Linux ships shims signed by both the old and the new CA to cover the transition. And their guidance includes a line that I think should be printed and taped to every machine in this building. Do not manually enroll the new KEK or CAs unless you know exactly what you're doing and have a tested recovery path. This can brick your system.



Corn

A distro telling you that you can brick your own machine by being helpful.



Herman

Because you can. The key database is not something the operating system owns. It's the firmware's, and you are reaching past the OS to change it.



Corn

And then the shim vulnerability, which is the other half of the current story.



Herman

ESET found eleven old Microsoft-signed shims, all version zero point nine or below, that can bypass Secure Boot on most UEFI machines. They lack MOK denylist enforcement, which arrived in zero point nine, and SBAT support, which arrived in fifteen point three. One Oracle Linux shim trusts a GRUB binary vulnerable to CVE twenty fifteen fifty two eighty one. Microsoft revoked all eleven in its June ninth patch Tuesday this year, tracked as CVE twenty twenty six eight eight six three and CVE twenty twenty six ten seven nine seven.



Corn

Eleven old signed shims sitting in the trust store. And the line from the reporting.



Herman

No one can say with confidence how many old, still-trusted shims remain. Which is the correct amount of confidence to have. Nobody has an inventory of every signed shim ever issued that's still in somebody's DB.



Corn

What's the reframe here, because I think we've been circling it.



Herman

Secure Boot is a key management problem. That's the whole thing. Not a technology problem, not a Microsoft problem, a key management problem. Expiring certificates, dual signed shims, new KEKs, the risk of bricking a machine by enrolling the wrong thing, eleven old shims nobody can count. That's all one category of problem, and it's the category that gets harder the more successful the feature becomes.



Corn

Every machine that trusts you is a machine you have to tell about the revocation.



Herman

And the revocation has to travel through firmware updates that may never come, on hardware whose owner has no idea any of this is happening.



Corn

Secure Boot is a signature check at launch. What is it actually checking. That's the chain itself, and I want it as roles rather than a diagram.



Herman

Roles. Firmware reads NVRAM and launches EFI applications. The EFI System Partition is FAT formatted and holds the bootloaders and often the kernels. The bootloader selects and loads. The kernel is the actual operating system. The initramfs is a cpio archive that provides early userspace, the modules, the udev rules, the tools, so the kernel can find and mount the real root filesystem. It's what handles LUKS, LVM, software RAID, network boot.



Corn

The initramfs is the piece that does the work nobody credits it for and gets blamed when it's wrong.



Herman

Every encrypted laptop that asks for a passphrase is asking because the initramfs got far enough to ask.



Corn

Now GRUB versus systemd-boot, because this is where the philosophical split lives.



Herman

GRUB does everything. BIOS and UEFI, MBR and GPT, built in support for RAID, LUKS and LVM, though not thin-provisioned volumes. systemd-boot is UEFI only, it launches only binaries from the ESP it's installed to or an XBOOTLDR partition on the same disk, and it auto-detects UKIs placed in the Linux directory on the ESP. That's it. That's the whole feature set.



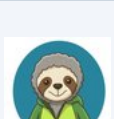
Corn

And the argument against GRUB now is not that it doesn't work. It's that it works too much.



Herman

It's feature-rich and it has been the source of a lot of CVEs. This year there's been an active push to strip it down. Julian Klode from Ubuntu Foundations argued in March that GRUB contains a lot of parsers for filesystems and other things which are a constant source of security issues. He proposed removing btrfs, hfsplus, xfs and zfs support, the jpeg and png decoders, and the Apple partition table parser from the signed GRUB builds in twenty six ten, keeping ext4, FAT, iso9660 and squashfs.



Corn

And the pushback, which is the honest tension.



Herman

Conan Kudo pushed back on the community side. Can we please not drop btrfs, because users who want boot-to-snapshot setups with Btrfs need that support. And on RAID, software RAID1 is incredibly common, dropping it would be a substantial loss.



Corn

So the fight is between the people who want the signed binary small enough to audit and the people who are actually using the features.



Herman

Which is the same fight as always. It's just sharper here because the thing being cut is inside the trust boundary.



Corn

Explain why that matters, because I think it's the key to the whole segment.



Herman

BootHole. CVE twenty twenty ten seven one three. A family of GRUB2 parser bugs that were exploitable through a legitimately signed GRUB binary. Nothing was forged. The binary was signed by the vendor, trusted by every firmware in the world, and it parsed attacker-controlled input, and that input could take over the machine. Microsoft had to push DBX revocations fleet-wide and the coordinated fix took over a year.



Corn

A signed binary that parses input from outside the trust boundary.



Herman

Every parser you leave in is attack surface inside the thing that's allowed to run before anything is verified. That's why the pressure exists. It's not that btrfs support is bad. It's that btrfs support is code, running as the most trusted thing on the machine, reading a filesystem image that an attacker might control.



Corn

And the structural answer to that is the UKI.



Herman

A Unified Kernel Image is a single PE binary. Same file format as a Windows executable. It bundles the kernel, the initramfs, the kernel command line, and a tiny stub bootloader called systemd-stub, into one file that UEFI can launch directly. No GRUB, no separate initrd, one binary with one signature.



Corn

One file, one hash, one signature to verify.



Herman

And that's what makes the next part work. Because it's a single signed binary, its PCR measurement is predictable. You can seal a LUKS key to PCR seven, which is the Secure Boot policy, and PCR eleven, which is the UKI hash, and the TPM will only release the key if both are what you measured. Tamper with the boot path, the PCR changes, the TPM declines, the key stays locked.



Corn

Which closes the initramfs hole from earlier.



Herman

That hole existed because the initramfs was this separate unsigned blob hanging off the end of a verified chain. Bundling it into the signed binary means it's covered by the signature and covered by the measurement. The simplification and the security fix are the same move.



Corn

So where does the distribution support actually stand, because I want to avoid selling a future that hasn't arrived.



Herman

It's uneven. Fedora ships UKIs by default for some installs, the Atomic editions and some server configurations. Arch has first-class UKI support since twenty twenty-three with systemd-boot recommended. Debian thirteen added UKI tooling, with Debian fourteen positioning to make it an installer option. RHEL ten supports them. OpenSUSE Tumbleweed has tooling. NixOS supports it through systemd-stub. Ubuntu supports UKI tooling since twenty four oh four LTS but the default install still uses GRUB and shim.



Corn

Ubuntu is not the leader here.



Herman

Ubuntu is deliberately not the leader here. They're stripping GRUB features rather than switching bootloaders, which is a defensible position when you're the distribution people dual-boot on twenty-year-old habits.



Corn

What are the costs, because there are always costs.



Herman

UKIs are eighty to a hundred and twenty megabytes. Some older UEFI implementations choke on EFI binaries that large. So the ESP guidance moves to two hundred and fifty megabytes minimum and five hundred if you want headroom, which is a lot for people who sized that partition at a hundred megabytes in twenty eighteen and never thought about it again.



Corn

The partition nobody resizes.



Herman

The partition nobody resizes and nobody can easily resize, because there's usually a Windows install on one side of it. TPM two point oh only, so machines without one are out. Firmware updates change PCR zero, which breaks sealed keys, so a BIOS update means re-enrolling. And UKIs are UEFI only, so the CSM has to be off, which loops back to the beginning of the episode.



Corn

And the distinction that gets conflated constantly.



Herman

Measured boot records what was loaded. It does not prevent loading. Only verified boot with enforcement prevents execution of unsigned code. Those get used interchangeably and they are not the same thing. A system measuring everything and enforcing nothing will tell you, after the fact, exactly what malware you booted.



Corn

Which is a log, not a lock.



Herman

A very good log.



Corn

So the Ubuntu LTS user on modern hardware. What do they actually need to understand.



Herman

If you install a current Ubuntu LTS on modern hardware and take the defaults, the default configuration is generally sufficient. That's the short answer. The things worth understanding are Secure Boot on or off, and the reason you'd turn it off is almost always a third-party driver. And whether your boot partition is encrypted.



Corn

Performance implications.



Herman

Essentially nil. That's not what this trades in. The security and reliability implications are real, but they're mostly handled by the default configuration. If you're turning Secure Boot off, you should know why, and it's usually because a module won't load. If you're leaving it on, you're leaning on a chain that doesn't check your initramfs, which the UKI work exists to fix.



Corn

I keep coming back to Klode's line about encrypted boot. That encryption of the boot partition only ever provided security by obscurity, not actual security. You want to ensure the integrity of those pieces.



Herman

Because encryption without integrity is only hiding. You can encrypt a boot partition and an attacker can still flip a byte inside it and you'll never know. Integrity is the thing you actually want, and that's what signatures and measurements provide.



Corn

Which is the whole arc of this episode. We spent twenty years encrypting things and we're now spending the next twenty proving that what got loaded is what we left there.



Herman

And doing that proving through key management, which is the hard part, which is where we came in.



Corn

Go back with me to the GRUB hardening fight for a second, because I think there's a version of it that's more interesting than the compatibility argument. If you're Ubuntu and you're the distro that a huge number of people dual-boot with, the correct move might be the conservative one. Strip the parsers, keep the bootloader, don't move anyone's boot path.



Herman

That's a real argument. The cost of a boot failure for Ubuntu is enormous because the install base is enormous and a large chunk of it is people who have no interest in re-learning a bootloader. Fedora and Arch can lead because their users are self-selected for wanting to.



Corn

The distro that breaks is the one that loses the user.



Herman

And you can't get that user back by explaining why the new way was better. They're already on a different distro.



Corn

I want to push on the Microsoft dependency, because I think there's a version of that concern that's overstated and a version that's fair. The overstated version is that Secure Boot is a Microsoft lock on Linux. That's not what's happening. The certificates just live in the DB and Microsoft got there first because Windows shipped on the hardware.



Herman

The fair version is the supply chain one. Every Linux shim in the world is signed by one company's infrastructure. If that infrastructure were compromised, or coerced, Secure Boot chains to attacker code. BootHole showed that a legitimately signed binary is enough to do real damage. That's a genuine structural tension, and it's not solved by disliking it.



Corn

And the counter is that the alternative is every OEM signing every distro, which would be worse.



Herman

Which is why nobody's seriously proposing it. The dependency is bad and the alternatives are worse, which is a very common shape for infrastructure problems.



Corn

One honest caveat on the obsolescence claim, because I don't want to overstate it. Intel's guidance is about validation. Validation is not the same as removal. Hardware can still have a CSM that isn't validated, and AMD's equivalent policy wasn't something I could pin down.



Herman

That's right, and it's worth saying. The strongest evidence is Intel's policy combined with the fact that they pushed partners to remove it by twenty twenty. Which means on any Intel platform built since then, it's gone. The practical answer to Daniel's question stands.



Corn

There's one thing I want to make sure we've said clearly, because it's the thing I got wrong for years. Legacy boot mode and the CSM are not old firmware. They're UEFI pretending to be old firmware, using a compatibility layer that was never in the spec.



Herman

Which is why the mental model of "UEFI is the new BIOS with options" fails. UEFI is a different architecture. The legacy path is a shim layered on top of it, and it's now being pulled out from underneath.



Corn

And the reason the distinction still confuses people who've been doing this for twenty years is that the words are used as a menu. BIOS or UEFI. Legacy or Secure Boot. Like they're two columns of the same settings screen.



Herman

When actually it's two different questions. Which firmware architecture are you on, and does your firmware enforce signature checks. You can be UEFI with enforcement off, and you can be UEFI with enforcement on but no PK enrolled, which is enforcement off with extra steps.



Corn

The menu lies to you.



Herman

The menu is a summary written by the people who made the machine, and they had to fit it in a box.



Corn

What does the twenty twenty-six CA transition actually mean for a normal user, though. Because I want to be honest about the blast radius.



Herman

The blast radius is small for anyone on hardware with current firmware and a distro that ships a dual-signed shim. The blast radius is real for anyone on hardware that stopped getting firmware updates before the new CAs were enrolled. That machine gets a shim



Herman (continued)

signed by the new CA, the firmware doesn't trust the new CA, and the boot fails.



Corn

And the fix requires the firmware to trust something it doesn't trust. Which means you need the old CA path or a manual enrollment.



Herman

Which is why Rocky's warning exists. If you're going to manually enroll the new KEK, you need a tested recovery path, because you can put the machine in a state where it won't boot anything.



Corn

A security feature that can brick your machine if you update it wrong, and can fail to boot if you update it not at all.



Herman

That's the operational cost, and it's the story, more than the certificate expiry.



Corn

I'm going to come back to the eleven shims, because the number is small and the meaning is large. Eleven signed binaries that can bypass Secure Boot, all issued years ago, all still trusted somewhere.



Herman

And the reason nobody can count how many remain is that trust is distributed. Every machine has its own copy of the DB and DBX. Microsoft can push a revocation, but there's no central inventory of who has received it.



Corn

The trust store is a distributed system nobody designed as one.



Herman

That's exactly what it is. It grew into being one, one firmware update at a time.



Corn

Which brings the lot back to the reframe. This is key management, and key management at scale is the part of security that nobody enjoys thinking about, because it's not a clever algorithm. It's a list of certificates and a process for removing them.



Herman

A process for removing them that has to travel through the one layer of the stack the operating system doesn't control.



Corn

The firmware.



Hilbert

Intel CSM Specification, version zero point nine eight.



Corn

...



Herman

That's the one we were just talking about.



Hilbert

Printed copy. Trade show, I don't want to say when. Man named Dale handed it to me. He had a laminated card that said Legacy Boot Liaison on it. I still have the card. The spec I gave to my brother in law, which was a mistake, because now he quotes it at me.



Corn

Of course he does.



Hilbert

Hosts are right about the CSM, by the way. Not in the UEFI spec. It was in Intel's framework, always was. But the reason legacy boot actually died isn't that it was badly designed. It's that Intel stopped validating it. And I have opinions about Intel's validation process that I am not going to get into on this show.



Herman

You can't just say that and stop.



Hilbert

I can. I'm producing. Herman, your levels drifted a little on that last line, pull the gain down a touch, there's a fan in the background. Anyway. The thing I actually wanted to say is about the MOK enrollment. I did one once. Physical confirmation at next boot, the firmware puts up a screen asking you to press a key, and I could not identify the key. So I pressed all of them.



Corn

All of them.



Hilbert

One at a time, working across the keyboard, and at some point the machine decided I had confirmed something and it booted into a firmware menu. In a language I did not recognise. I have never worked out which key it was. I still have the machine. It still boots. I use it to run one program and I am not going to tell you what the program is.



Herman

Hilbert.



Hilbert

It's a bootloader. Cost me nothing, it came on a floppy somebody left at the studio, and it does one thing. My brother in law says it's dangerous. My brother in law also says the CSM spec I gave him was the wrong version, and he is not to be trusted on either point.



Corn

The machine you enrolled a key on by pressing every key on the keyboard is still booting.



Hilbert

Every morning. Boots faster than the new one.



Herman

Do you still have the key?



Hilbert

Different question. I've got to cut a cough out of my own track from ten minutes ago, so I'll leave you with this. Intel didn't remove legacy boot because it was insecure. They removed it because they stopped testing it. Those are different reasons and only one of them is about you.



Corn

Which is the actual answer to Daniel's question, in a way.



Herman

It is. Obsolescence is a maintenance decision, not a technical verdict.



Corn

The boot stack is converging on UEFI, then systemd-boot, then a UKI. That replaces UEFI, then shim, then GRUB, then kernel plus initrd. One signed binary, one measurement, one thing to trust.



Herman

The open question is whether that's better for users or whether it just moves the trust problem somewhere smaller and harder to see. Because a single binary measured by the TPM is cleaner to reason about, and it's also one artifact that either works or doesn't.



Corn

The transition is happening right now. A machine that stopped getting firmware updates before the new CAs were enrolled will fail to boot a shim signed only by the old one. That's not hypothetical, that's hardware sitting in people's houses today.



Herman

The security benefit of all this is real. The operational cost is the story, and the cost is key management, and key management is hard. That's where I'd leave it.



Corn

Thanks to our producer Hilbert Flumingtop, who is currently cutting a cough out of his own track.



Herman

If you want more of this, try episode thirteen seventeen, The First Second; episode eleven, How Does Fine Tuning Work Anyway; and episode seven twenty-nine, The Surprising Family Tree of Modern Operating Systems. This has been My Weird Prompts.



Corn

If you want to send us your own prompt, send it on Telegram at t dot me slash MWP listener bot. We'll be back soon.

SOURCES

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

1. **ArchWiki [primary]**
   Arch boot process (firmware types, CSM, bootloader comparison, initramfs)
   https://wiki.archlinux.org/title/Arch_boot_process

2. **ArchWiki [primary]**
   Unified Extensible Firmware Interface (UEFI variables, efibootmgr, fallback paths)
   https://wiki.archlinux.org/title/Unified_Extensible_Firmware_Interface

3. **ArchWiki [primary]**
   Unified kernel image (UKI structure, signing, booting)
   https://wiki.archlinux.org/title/Unified_kernel_image

4. **Ubuntu security documentation [primary]**
   UEFI Secure Boot (last updated 2026-01-22)
   https://documentation.ubuntu.com/security/security-features/platform-protections/secure-boot/

5. **Ubuntu release notes [primary]**
   26.04 LTS (2026-04-23)
   https://documentation.ubuntu.com/release-notes/26.04/

6. **Ubuntu Discourse [primary]**
   Streamlining secure boot for 26.10 (Julian Klode, 2026-03-25)
   https://discourse.ubuntu.com/t/streamlining-secure-boot-for-26-10/79069

7. **Rocky Linux**
   Rocky Linux and the changing Secure Boot landscape (2026-06-02)
   https://rockylinux.org/news/2026-06-02-secure-boot-ca-kek-transition

8. **Help Net Security**
   No one knows how many old shims can still bypass UEFI Secure Boot (2026-07-14)
   https://www.helpnetsecurity.com/2026/07/14/eset-uefi-secure-boot-bypass/

9. **Big Iron**
   Unified Kernel Images (UKIs) and shim replacement in 2026 (2026-06-12)
   https://www.bigiron.cc/guides/shim-replacement-and-uki-unified-kernel-images-2026

10. **systemshardening.com**
UEFI Secure Boot Deep Dive: DB/DBX, Shim, MOK (2026-05-07)
https://www.systemshardening.com/articles/linux/linux-uefi-secure-boot-db/

11. **Intel**
Removal of Legacy Boot Support for Intel Platforms
https://cdrdv2.intel.com/v1/dl/getContent/630266

12. **TechPowerUp**
Intel to Remove Legacy BIOS Support from Motherboard UEFI in 2020
https://www.techpowerup.com/238950/

13. **SunbeamTech**
CSM Support in BIOS: On or Off (CSM not part of UEFI spec)
https://sunbeamtech.com/bios-settings/csm-bios-setting-explained/

14. **Progressive Robot**
Ubuntu 26.04 LTS, Secure Boot blocks unsigned DKMS kernel module load (2026-07-05)
https://www.progressiverobot.com/2026/07/05/ubuntu-26-04-lts-secure-boot-blocks-unsigned-dkms-kernel-module-load-fix-prevention/

15. **Hacker News**
Tell HN: Microsoft SecureBoot 'Breaking' Changes (2024-08-14, SBAT context)
https://news.ycombinator.com/item?id=41242943