



UEFI, HP, and Me: A Bootloader Love-Hate Story

How one stubborn laptop taught me more about firmware psychology
than I ever wanted to know

This is a story about this lovely old HP business laptop and its
“interpretation” of the UEFI specification -- “Quirky” would be
generous.

Demystifying Bootloaders

Term	Meaning	Reality
BIOS	<i>Basic Input/Output System</i> : A ghost of the original IBM PC/XT	Bodges layered upon bodes; boots your hot new PC as if it was fresh out of 1983
UEFI	<i>Unified Extensible Firmware Interface</i> : The modern replacement	A fresh take; smarter, flexible, but still occasionally cursed
CSM	<i>Compatibility Support Module</i> : A component of UEFI that allows legacy booting (a.k.a BIOS mode)	Usually throws away UEFI's ability to recognize a GPT disk set up for legacy booting, falling back to BIOS heuristics for determining bootability
MBR	<i>Master Boot Record</i> : A 512-byte relic that supports, without hacks, a maximum of 4 partitions	Designed when a "large drive" was 10MB; yet somehow still alive
GPT	<i>GUID Partition Table</i> : Modern, robust, and sanity-preserving	The only sane partitioning scheme for 2025; theoretically supports BIOS booting

There are two ways you can boot a modern PC.

You can pretend it's still 1983 and boot your super-duper **Ryzen**

Threadripper 7995WX exactly like it's a **4.77 MHz Intel 8088**.

BIOS (or **CSM**) will dutifully set up a handful of abstraction layers (in 16-bit real mode, mind you) to make your modern computer look like an IBM 5150, load the first 512-byte sector of your **8 TiB NVMe SSD** into a sacred address somewhere in the first 640 kiB of your **128 GiB of DDR5**; then plunge head-first into it – still in real mode, still limited to a 64 kiB code segment – and hope for the best.

Or, you can do it the sane way; with **UEFI**, which actually understands partition tables and filesystems, natively supports USB and PCIe, can load drivers, draw graphics, run applications, and directly boot your operating system kernel with the CPU already in **64-bit protected mode** where it belongs.

Naturally, UEFI works equally well with both MBR and the preferred GPT partitioning scheme.

BIOS works best with MBR-partitioned disks, but can be coerced into booting GPT-partitioned disks. However, actually doing so is akin to firmware necromancy – and best avoided unless you *really* enjoy debugging ancient rituals; as I was to discover when I started poking at Haiku for my previous talk. But, more on that later.

Meet the Stubborn Hardware

- HP Elitebook 8560p (circa 2011)
- Early Insyde-based EFI firmware
- EFI support primarily intended for internal consumption (e.g. diagnostics and updates)
- Dual-drive setup: Crucial MX200 mSATA SSD (Expansion bay) + original WD Black HDD, both 500 GB
- Mission: A modern, reliable boot into any OS



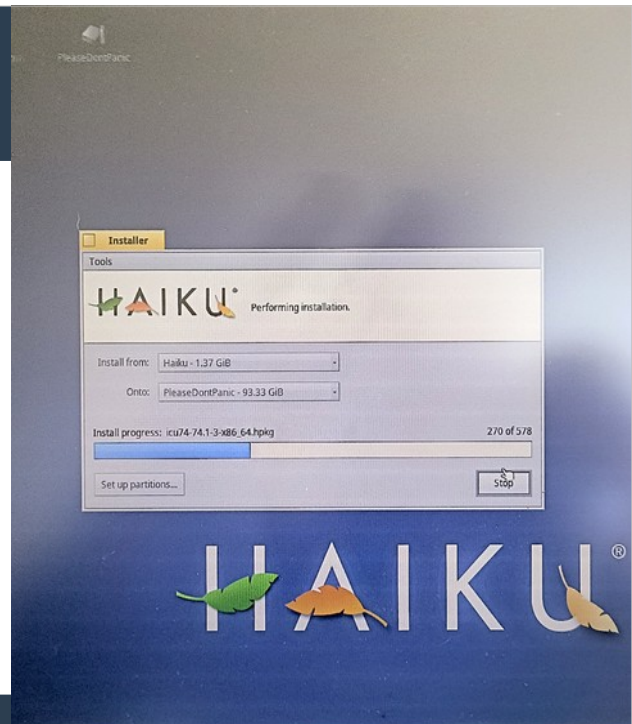
This laptop comes from that awkward moment in history where OEMs were only just coming to terms with EFI on the PC platform – Of course, HP had been doing it for *years* in the server space; they really had no excuse for the horrors we are about to witness.

Regardless, UEFI of this time was mostly used for *vendor tooling*; it wasn't actually expected that end-users would *use* it.

Aside: The astute among you may ask; why mount the SSD in the expansion bay (where it is throttled to SATA II), when it could be running SATA III in the HDD bay? That's a reasonable question, and it originally came down to ease of mounting; but I have since rectified this with a custom mounting bracket.

The Catalyst

- GPT disks booted in legacy mode
- Fragile BIOSBOOT shim prone to breakage
- GRUB forgot how to boot Haiku (after convincing it to reinstall)
- How hard can it possibly be to boot UEFI?



Legacy booting of GPT disks is *technically* supported; and there is no sane reason to use MBR partitioning in 2025; so that is exactly what I did.

It would've worked, if it weren't for those meddling partition managers. You see, the majority of late pre-EFI BIOSes (and, apparently, some early UEFIs that really should know better) will refuse to boot a disk unless its MBR partition table – which lives on as part of the GPT header – has the “boot” flag set.

Now, in theory, this is fine. GPT specifies a “BIOS BOOT” partition type specifically for this purpose, and the boot flag in the MBR can redirect boot to that just fine. And **grub2-install** will even set things up this way if it detects the appropriate partition layout.

Unfortunately, on paper, virtually every GPT partitioning tool out there completely ignores the BIOS BOOT partition and nukes the “boot” flag from the protective MBR; rendering the disk unbootable. There is a workaround for this, of course – “Hybrid MBR”; which is even more fragile than it sounds.

So, there's really only one option here. I need to convince the firmware to boot its modern disks using modern UEFI.

"Sure, I can help you with that! Absolutely!"

Holding Shift usually shows the GRUB menu, except when it doesn't; I should probably just reconfigure GRUB to show the menu unconditionally; or (better) find a way to convince Hewlett Packard's half-baked UEFI implementation to boot all the OSes I want. That'd be a lot better than relying on fragile legacy booting from a GPT-formatted disk.

Exactly — the root of your trouble here isn't really GRUB at all, it's the awkward handshake between HP's quirky UEFI and your multi-boot GPT setup.

Right now you're stacking two layers of fragility:

- Legacy BIOS boot from GPT (which is already a "this works until it doesn't" situation),
- GRUB's "only show menu when it feels like it" default.

A more stable setup would be:

1. Switch to proper UEFI boot for all OSes

- Even if HP's UEFI is flaky, you can usually tame it with a combination of:
 - Setting `BootOrder` with `efibootmgr` (Linux) so Fedora, Haiku, etc., are all first-class citizens.
 - Ensuring each OS has its `.efi` loader in `\EFI\<OS>` plus a copy in `\EFI\Boot\bootx64.efi` as a fallback ↓

2. Install GRUB in EFI mode (not legacy)

I did have rEFInd installed on there. The complication seems to be that HP doesn't properly support EFI variables; which was leading to the installers of _some_ OSes to fail when trying to install the bootloader; even though the installation process was largely successful.

Ah — yep, that's the classic HP "UEFI but only in the most begrudging sense" problem.

On a lot of EliteBooks from that era, the firmware will:

- Pretend to support EFI variables, but lose them after a reboot or firmware update.
- Ignore `efibootmgr` changes unless they target the fallback path (`\EFI\Boot\bootx64.efi`).
- Sometimes hardcode its boot to **only** load `bootx64.efi` from the EFI System Partition (ESP) — no matter what `BootOrder` says.

That explains why rEFInd worked for you — it doesn't *need* EFI variable support if you just plant it in the fallback location.

A trick that usually works on stubborn HPs is:

1. Put your preferred boot manager (rEFInd or GRUB EFI) as the fallback loader

- Copy it to:

```
bash
/EFI/Boot/bootx64.efi
```

Copy code

- This is the "if all else fails" file the firmware *will* run.

So, I asked GPT-5 for advice. It seemed like the right thing to do; considering it had just finished a post-mortem on my Haiku install; which boiled down to a corrupt BFS partition and a BFS formatting tool saying "I made a new filesystem; she's all good, bro" while ignoring the leftover landmines the installer and its kernel was repeatedly tripping over.

This began a journey of mutual discovery – mostly discovering how badly HP can engineer UEFI; and had the side-effect of actually documenting the journey so I can talk about it today.

It also birthed a several-month-long side project just so I can edit **refind.conf**; but that's a topic for another day.

The Observation

- Scary warning when enabling UEFI
- Installing OS in UEFI mostly works, but can fail when setting bootvars
- HP ignores bootvars that are set
- CSM cannot be disabled
- “Boot From EFI File” is the only reliable way to EFI boot external media

```
Make sure Secure Boot is disabled before proceeding.  
root@silverfish:~# grub2-install --target=x86_64-efi --efi-directory=/boot/efi --bootlo  
Installing for x86_64-efi platform.  
Could not prepare Boot variable: No space left on device  
grub2-install: error: efibootmgr failed to register the boot entry: Input/output error.  
root@silverfish:~#
```

UEFI Boot Mode

Warning

The “UEFI Boot” option on this system is provided for development purposes only and is currently NOT fully supported or warranted by HP.
Warning

The “UEFI Boot” option on this system is provided for development purposes only and is currently NOT fully supported or warranted by HP.
Preboot Authentication and Drive Lock are currently NOT supported under the UEFI Boot mode.
HP strongly recommends disabling Preboot Authentication and Drive Lock before enabling UEFI Boot on this system.

Accept
Cancel

Please Select

Notebook Upgrade Bay (UEFI)

Notebook Ethernet (UEFI)

OS Boot Manager

Boot From EFI File

Notebook Hard Drive

Notebook Upgrade Bay

USB Hard Drive 1 - TDKMedia Platinum 3.0

It's EFI, Jim; but not as we know it. Intimidating warnings, mysterious OS installation failures, and a firmware that goes all in on pretending it's nothing more than a highly evolved BIOS wearing an EFI costume. It gestures enthusiastically toward UEFI, while fundamentally refusing to cooperate.

The Plan

- Boot Fedora in EFI mode using SystemRescueCD
- Install rEFInd and an EFI bootloader
- ͸(͡ ͡)_
- Profit

```

:: Triggering events...
:: running hook [acndisk]
:: running hook [findroot]
Searching for block devices ...
=====
NAME      SIZE FSTYPE LABEL
-----
/dev/sda  465.8G
/dev/sda1  1M
/dev/sda2  1G ext4
/dev/sda3  195.3G ntfs    Fedora_silverfish09
/dev/sda4  93.3G btrfs    PleaseDon'tPanic
/dev/sda5  5G  ufat     HP_TOOLS
/dev/sdb  465.8G
/dev/sdb1  100M ufat
/dev/sdb2  16M
/dev/sdb3  465.1G ntfs
/dev/sdb4  539M ntfs
/dev/sdc  29.4G
/dev/sdc1  29.4G ufat
/dev/sdc2  32M ufat  UDTVEFI
=====
Checking for /sbin/init on device /dev/sda ...
Checking for /sbin/init on device /dev/sda1 ...
Checking for /sbin/init on device /dev/sda2 ...
Checking for /sbin/init on device /dev/sda3 ...
Checking for /sbin/init on device /dev/sda4 ...
[ 19.495601] befs: (sda4): ag_shift disagrees with blocks_per_ag.
[ 19.497647] befs: (sda4): <--- befs_find_key sbin not found
[ 19.497707] befs: (sda4): <--- befs_btrfs_find key sbin not found
Checking for /sbin/init on device /dev/sda5 ...
Checking for /sbin/init on device /dev/sdb ...
[ 19.743482] EXT4-fs (sdb): UFS: Can't find ext4 filesystem
[ 19.743787] EXT4-fs (sdb): UFS: Can't find ext4 filesystem
[ 19.743882] EXT4-fs (sdb): UFS: Can't find ext4 filesystem
[ 19.744042] befs: (sdb): invalid magic header
[ 19.744210] FAT-fs (sdb): bogus number of reserved sectors
Checking for /sbin/init on device /dev/sdb1 ...
[ 19.762664] EXT4-fs (sdb2): UFS: Can't find ext4 filesystem
[ 19.762839] EXT4-fs (sdb2): UFS: Can't find ext4 filesystem
[ 19.762994] EXT4-fs (sdb2): UFS: Can't find ext4 filesystem
[ 19.763149] befs: (sdb2): invalid magic header
[ 19.763298] FAT-fs (sdb2): bogus number of reserved sectors
Checking for /sbin/init on device /dev/sdb3 ...
Checking for /sbin/init on device /dev/sdb4 ...
[ 19.871131] EXT4-fs (sdc): UFS: Can't find ext4 filesystem
[ 19.872442] EXT4-fs (sdc): UFS: Can't find ext4 filesystem
[ 19.873571] EXT4-fs (sdc): UFS: Can't find ext4 filesystem
[ 19.874731] befs: (sdc): invalid magic header
[ 19.875925] FAT-fs (sdc): bogus number of reserved sectors
Checking for /sbin/init on device /dev/sdc1 ...
Checking for /sbin/init on device /dev/sdc2 ...
ERROR: Failed to find /sbin/init on any block device, cannot continue
sh: can't access tty: job control turned off
rootfs "ja _

```

The plan was simple. Use a rescue disk to boot my Fedora installation in EFI mode, install its favoured EFI bootloader (GRUB? Systemd-Boot? EFI-stub? I wasn't going to be picky), along with rEFInd as a nice boot selector, *shrug*, and profit.

So, I found a rescue OS, and dived headfirst into attempting to boot Fedora in EFI mode so I could install an EFI bootloader for it. This went about as well as expected.

The Reality

- After much ado, Wayland boots
- But no mouse, no WiFi
- Switch to VT, realise half of PCI devices missing, fall back to wired Ethernet
- Download & install GRUB EFI and rEFInd



```
root@silverfish:~# grub2-install --target=x86_64-efi --efi-directory=/boot/efi --bootloader-id=Fedora
Installing for x86_64-efi platform.
grub2-install: error: This utility should not be used for EFI platforms because it does not support UEFI Secure Boot. If y
Make sure Secure Boot is disabled before proceeding.
root@silverfish:~# grub2-install --target=x86_64-efi --efi-directory=/boot/efi --bootloader-id=Fedora --force
Installing for x86_64-efi platform.
Could not prepare Boot variable: No space left on device
grub2-install: error: efibootmgr failed to register the boot entry: Input/output error.
root@silverfish:~#
```

Regardless, I eventually stumbled upon the magic incantation to mount Fedora's root inside SystemRescue's initramfs and boot into it.

Eventually, I was greeted with SDDM on Wayland, and I thought it would be smooth sailing from here. I was wrong.

Fortunately, SystemRescue's kernel has an Intel 82579LM driver that works. And fortunately, I have this little D-Link router I routinely use as a reverse WiFi access point. So, I proceed to install GRUB and rEFInd to the mounted Windows ESP on the SSD. Or, so I thought...

Bootloader Hide-and-Seek

- Dude, where's my bootloader?
- Dude, *there's* your bootloader
- How did it get there, Dude?
- Dude, we were so messed up last boot

```

[root@sysrescue ~]# rm -rf /mnt/sdb1/EFI/{Boot,Fedora}
[root@sysrescue ~]# mv /mnt/sda2/efi/EFI /mnt/sdb1/
mv: inter-device move failed: '/mnt/sda2/efi/EFI' to '/mnt/sdb1/
[root@sysrescue ~]# mv /mnt/sda2/efi/EFI/* /mnt/sdb1/EFI/
[root@sysrescue ~]# ls /mnt/sda2/efi /mnt/sdb1/EFI
/mnt/sda2/efi:
EFI mach_kernel System

/mnt/sdb1/EFI:
BOOT fedora Microsoft refind tools

```

```

/dev/sdb1 on /boot/efi type vfat (rw,relatime)
/dev/sda2 on /boot type ext4 (rw,relatime)
root@silverfish:~# umount /dev/sda1
umount: /dev/sda1: not mounted.
root@silverfish:~# umount /dev/sdb1
umount: /boot/efi: not mounted.
root@silverfish:~#

```

```

/mnt/sdb1/EFI:
Boot Fedora Microsoft

```

```

[root@sysrescue ~]# ls /mnt/sda2/efi/
EFI/          mach_kernel System/
[root@sysrescue ~]# ls /mnt/sda2/efi/EFI
BOOT fedora refind tools
[root@sysrescue ~]# _

```

As it turned out, “mount” had completely lost its sanity and failed to actually mount the ESP; so all those boot files ended up in a literal “EFI” directory inside Fedora’s ext4 boot partition. So I put them where they belong and rebooted.

Firmware Forensics

```

rEFInd - Booting OS
GRUB version 2.12
Minimal BIOS-like line editing is supported. For the first word, TAB lists possible command completions. Any
TAB lists possible device or file completions. To enable less(1)-like paging, "set pager=1".
thing load options ""
grub> _

```

```

Checking for /sbin/init on device /dev/sdb1 ...
Checking for /sbin/init on device /dev/sdb2 ...
[ 19.532579] EXT4-fs (sdb2): UFS: Can't find ext4 filesystem
[ 19.532833] EXT4-fs (sdb2): UFS: Can't find ext4 filesystem
[ 19.532990] EXT4-fs (sdb2): UFS: Can't find ext4 filesystem
[ 19.533151] befs: (sdb2): invalid magic header
[ 19.533301] FAT-fs (sdb2): bogus number of reserved sectors
Checking for /sbin/init on device /dev/sdb3 ...
Checking for /sbin/init on device /dev/sdb4 ...
Checking for /sbin/init on device /dev/sdc ...
[ 19.635176] EXT4-fs (sdc): UFS: Can't find ext4 filesystem
[ 19.635779] EXT4-fs (sdc): UFS: Can't find ext4 filesystem
[ 19.636331] EXT4-fs (sdc): UFS: Can't find ext4 filesystem
[ 19.636997] befs: (sdc): invalid magic header
[ 19.637550] FAT-fs (sdc): bogus number of reserved sectors
Checking for /sbin/init on device /dev/sdc1 ...
Checking for /sbin/init on device /dev/sdc2 ...
ERROR: Failed to find /sbin/init on any block device, cannot continue
sh: can't access tty: job control turned off
[rooftfs]# mount -o subvol=root /dev/sda3 /new_root
[rooftfs]# exec switch-root /new_root /sbin/init
sh: exec: line 2: switch-root: not found
[ 57.098200] Kernel panic - not syncing: Attempted to kill init! exitcode=0
[ 57.098339] CPU: 1 UID: 0 PID: 1 Comm: sh Not tainted 6.12.30-1-lts.el8
[ 57.100499] Hardware name: Hewlett-Packard HP EliteBook 8560p/1618, BIOS
[ 57.101663] Call Trace:
[ 57.102829] <TASK>
[ 57.104012] dump_stack_lul+0x5d/0x80
[ 57.105109] panic+0x110/0x22b
[ 57.106359] do_exit.cold+0x15/0x58
[ 57.107527] do_group_exit+0x2d/0xc0
[ 57.108685] __x64_sys_exit_group+0x18/0x20
[ 57.109842] x64_sys_call+0x14b4/0x14c0
[ 57.111009] do_syscall_64+0x7b/0x190
[ 57.112175] ? syscall_exit_to_user_mode+0x37/0x1c0
[ 57.113358] ? do_syscall_64+0x87/0x190
[ 57.114539] ? syscall_exit_to_user_mode+0x37/0x1c0
[ 57.115726] ? do_syscall_64+0x87/0x190
[ 57.116901] ? __x64_sys_rt_sigaction+0x11b/0x140
[ 57.118169] ? syscall_exit_to_user_mode+0x37/0x1c0
[ 57.119349] ? do_syscall_64+0x87/0x190
[ 57.120513] ? irqentry_exit_to_user_mode+0x2c/0x1b0
[ 57.121699] entry_SYSCALL_64_after_hwframe+0x7b/0x7e
[ 57.122901] RIP: 0033/0x7594dc89160
[ 57.124101] Code: ff 64 09 02 eb c4 67 eb c5 64 1d 04 00 66 0f 1f 44 00 00
if 04 00 00 00
[ 57.127060] RSP: 002b:00007ffc3a0e3c18 EFLAGS: 00000202 ORIG_RAX: 00000000
[ 57.128351] RAX: ffffffff/fffffda RDI: 00000000/00000000 RSI: 00007594dc89160
[ 57.130220] RDX: 00000000/00000000 R0: ffffffff/fffffda RDI: 00000000/00000000
[ 57.131807] RBP: 00000000/00000000 R08: 00007ffc3a0e3c18 R09: 00007ffc3a0e3c18
[ 57.133415] R10: 00000000/00000000 R11: 00000000/00000000 R12: 00000000/00000000
[ 57.135026] R13: 00000000/00000000 R14: 00007594dc89160 R15: 00000000/00000000
[ 57.136665] </TASK>
[ 57.138374] Kernel Offset: 0x9e00000 from 0xfffffda10000000 (relocation)
[ 57.139451] ---[ end Kernel panic - not syncing: Attempted to kill init

```

- EFI Bootvars are either ignored or broken
- Install rEFInd to fallback path (<ssd>/EFI/BOOT/BOOTX64.EFI)
- Set bootorder "Expansion Bay (UEFI)" then "OS Boot Manager"
- HP tries to boot rEFInd, fails, reboots into Windows on the *same ESP*
- Manually booting rEFInd works

After *definitely* not forgetting to regenerate grub.cfg, hijacking SystemRescue's initramfs to boot back into Fedora, and installing GRUB *properly*; it was time to dive deeper down the rabbit hole. rEFInd and Fedora now boot; but only if I directly point the firmware to /EFI/BOOT/BOOTX64.EFI on the SSD's ESP.

But, I ain't going to use the equivalent of a "File→Open" dialogue to boot an operating system – that's insane!

HP_TOOLS Experiment (Pt. 1)

- **Try to repurpose “HP_TOOLS” at end of HDD as ESP**
 - Install rEFInd to “/EFI/BOOT/BOOTX64.EFI”
 - Firmware ignores it (not in bootorder selection) and continues booting Windows
- **Use partitioner to “bless” HP_TOOLS as a real ESP**
 - Firmware still stubbornly refuses to acknowledge it
- **Shrink from 5GB to 500MB?**
 - No change in behaviour
- **Rename from “HP_TOOLS” to “ESP_HPTOOLS”?**
 - Completely indifferent

```

uefi - C12A7328-F81F-11D2-BA4B-00A0C93EC93B
raid - A1D0880F-08FC-4D3B-A006-743F0F84911E
lvm - E0000370-F507-44C2-A23C-238F2A3DF928
xbootldr - BC13C2FF-59E6-4262-A952-B275F06F7172
Partition type or alias (type L to list all): L

Changed type of partition 'Microsoft basic data' to 'EFI System'.

Command (m for help): p
Disk /dev/sda: 465.76 GiB, 500107862016 bytes, 976773168 sectors
Disk model: WDC WD5000BBPKT-6
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes
Disklabel type: gpt
Disk identifier: EDA7AF03-146C-4152-9364-178258A72AE7

Device      Start      End      Sectors  Size Type
/dev/sda1    2048      4095      2048      1M BIOS boot
/dev/sda2    4096    2101247    2097152    1G Linux extended boot
/dev/sda3    2101248  41701247  40600000    195.3G Linux filesystem
/dev/sda4    411701248 412725247  1024000     500M EFI System

Command (m for help): L 4
Partition number (1-4, default 4): 4

Device: /dev/sda4
Start: 411701248
End: 412725247
Sectors: 1024000
Size: 500M
Type: EFI System
Type-UUID: C12A7328-F81F-11D2-BA4B-00A0C93EC93B
UUID: 5E92EEF9-B5DE-4923-902C-830E01FAA71D
Name: EFI System Partition

Command (m for help): w
The partition table has been altered.
Syncing disks.

```

The Windows utility I originally used to create the “HP_TOOLS” partition and install the firmware update and diagnostics decided it should be 5GB. Why? Who knows. So, I figured I might as well try using it as an ESP. This was met with *some resistance*.

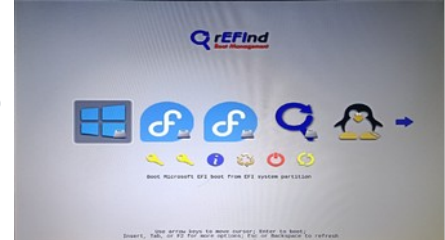
So, I used fdisk and parted to “bless” it as a System ESP. Still no.

Let’s shrink it. Maybe it’s confused by a 5GB ESP? No.

Maybe it’s refusing because of the filesystem label? No.

HP_TOOLS Experiment (Pt. 2)

- **Rename “HP_TOOLS” to “ESP_HPTOOLS”**
 - Breaks connection to HP diagnostics
 - Still refuses to enumerate HDD as an EFI boot candidate
- **Copy ESP contents from SSD into “ESP_HPTOOLS”, pull SSD**
 - Firmware *still* refuses to acknowledge it as boot option
 - No problem finding Windows Boot Manager, though
- **Rename “ESP_HPTOOLS” back to “HP_TOOLS”**
 - Same result, but tools work
- **Remove Windows Boot Manager from “HP_TOOLS”, masquerade rEFInd as bootmgfw.efi**
 - *Finally*, rEFInd boots by default
- **Chaos theory: Reinstall SSD to expansion bay, with authentic ESP and Windows bootloader**



Renaming the partition broke its connection to HP diagnostics, but otherwise had no effect on the firmware’s boot behaviour.

So I copied everything from the SSD’s ESP onto the HP_TOOLS partition, removed the expansion bay, and tried again.

This was when GPT and I came to the painful realisation that HP’s firmware was really only ever looking for *one* thing; and it was neither EFI boot variables nor standard EFI fallback paths.

So, I begrudgingly installed rEFInd as “/EFI/Microsoft/Boot/bootmgfw.efi” and cursed VFAT for not letting me symlink to it. *Victory by deception.*

Current Situation

- **“HP_TOOLS” resides as first partition of first disk, as an ordinary “msftdata” partition**
 - Contains rEFInd at “/EFI/Microsoft/Boot/bootmgfw.efi”
 - Contains HP Tools at “/Hewlett-Packard”
- **“EFI_SYS” is next partition, tagged as real ESP**
 - Contains all OS bootloaders, including Windows
- **Firmware doesn’t care about special flags or partition names; just boots the first “bootmgfw.efi” from the first FAT32 partition it sees**

The most cursed thing about all of this? HP literally doesn’t care about the partition type; well, at least, not on GPT. If it’s a FAT32 partition, and it contains /EFI/Microsoft/Boot/bootmgfw.efi; then it’s booting it, no questions asked.

So, now I have:

- * A perfectly ordinary, innocuous-looking data partition that’s secretly an ESP with rEFInd as a trojan horse.
- * Next door to that, a real ESP, containing real bootloaders for real OSes
- * A UEFI-ish firmware that is blissfully unaware of all the chaos it has created.

I *wish* that was the end of the story; but the worst is yet to come.

WiFi Woes

Following a recent kernel update, my Centrino WiFi card decided to destabilize, fall off the PCI bus, and take down half the kernel with it. So, I decided it was time for an upgrade.

I did my due diligence; and selected a new-old-stock Intel 7260HMW; with a genuine HP OEM part number on it; from PB-Tech. An original HP replacement part for an original HP laptop. Couldn't get any simpler, right?

HP Firmware – The Gift that Keeps on Giving

- *Of course* HP's 2018 firmware rejects HP's own OEM-approved genuine replacement part, on sale since 2013
- Because, why wouldn't it?
- Why would anybody want a 7260HMW when they can have a Centrino 6205 with broken drivers?



Wrong. Hewlett Packard had *seven years* to either add this module to the firmware's whitelist (which would've been reasonable, since both products were on the market at the same time), or even to just disable the whitelist when they discontinued support for the laptop (the 2018 firmware was released *long* after this model was sunset). Instead, they doubled down. They chose *violence*.

And it's one thing to show a nag screen on boot; but to power-gate the slot so the OS can't even see it? That's just *petty*. So now I have a perfectly acceptable, perfectly capable mini PCI express WiFi 5 + BT 4 card that isn't even useful as a paperweight. *Thanks, HP!*

But, throughout this journey, we haven't even answered the *most important* question yet.

But, Can it Run DOOM?



Can it run DOOM? Well, *can it?*

But, Can it Run DOOM?

- **No. Even demons tremor at the horrors of HP's mutilated Insyde BIOS**
- **HP's priority isn't booting your OS**
- **Bob wants a Bonus; so bundle in:**
 - An offline PIM EFI app (QuickLook)
 - An instant-on OS (QuickWeb, aka SplashTop)
 - An SMM calendar that replaces the Windows boot screen (DayStarter)

```

EFI Shell version 2.00 (4096.1)
Current running mode 1.1.2
Device mapping table
fs0 :HardDisk - Alias hd29a0a1 blk0
      \PciRoot(0x0)/Pci(0x1F,0x2)/?/HD(1,GPT,409C63B2-46B0-4CB9-A21F-618E20
fs1 :HardDisk - Alias hd29a0a2 blk1
      \PciRoot(0x0)/Pci(0x1F,0x2)/?/HD(2,GPT,3077109D-E727-489C-B1B9-EC97A6
blk0 :HardDisk - Alias hd29a0a1 fs0
      \PciRoot(0x0)/Pci(0x1F,0x2)/?/HD(1,GPT,409C63B2-46B0-4CB9-A21F-618E20
blk1 :HardDisk - Alias hd29a0a2 fs1
      \PciRoot(0x0)/Pci(0x1F,0x2)/?/HD(2,GPT,3077109D-E727-489C-B1B9-EC97A6
blk2 :HardDisk - Alias (null)
      \PciRoot(0x0)/Pci(0x1F,0x2)/?/HD(3,GPT,32D2A698-D696-4947-AD55-055060
blk3 :HardDisk - Alias (null)
      \PciRoot(0x0)/Pci(0x1F,0x2)/?/HD(1,GPT,2C2DE476-1F50-42B9-8D2E-D9B6AC
blk4 :HardDisk - Alias (null)
      \PciRoot(0x0)/Pci(0x1F,0x2)/?/HD(3,GPT,5E8FBFC2-E4CF-40A7-81AE-66F676
blk5 :HardDisk - Alias (null)
      \PciRoot(0x0)/Pci(0x1F,0x2)/?/HD(4,GPT,6A426A5A-DB58-4E42-B1FE-FDFC0A
blk6 :BlockDevice - Alias (null)
      \PciRoot(0x0)/Pci(0x1F,0x2)/?
blk7 :BlockDevice - Alias (null)
      \PciRoot(0x0)/Pci(0x1F,0x2)/?

Press ESC in 1 seconds to skip startup.nsh, any other key to continue.
Shell> blk0:
blk0:> cd \EFI\tools
blk0:\EFI\tools> doom.efi
./doom1.wad

Arguments: doom.efi.
DOOM Shareware Startup v1.10
U_Init: allocate screens.
M_LoadDefaults: Load system defaults.
Z_Init: Init zone memory allocation daemon.
U_Init: Init WADfiles.
adding ./doom1.wad

=====
Shareware!
=====
M_Init: Init miscellaneous info.
R_Init: Init DOOM refresh daemon -
R_Init:

```

DOOM runs everywhere; or, at least, that's what the meme suggests. But, you see, HP's priority was not to implement UEFI in any way that is actually useful to end users. If it can multitask a Personal Information Manager in System Management Mode while simultaneously booting Windows 7 in CSM mode; then that's "Enterprise Ready" and Bob gets his bonus. Obviously, a firmware should *not* be trusted to manage your hardware if it can't even manage your *schedule*.

DOOM died the moment it tried to draw a pixel. When prompted, the AI was unable to determine if this is a bug or a feature.

Thank You!

**NO! I
WILL NOT
EFI BOOT
LINUX!**



**AND YOU
CAN'T
MAKE ME!**

So there you have it.

After about a week of negotiation, firmware psychology, and light identity fraud...

my EliteBook finally boots like a normal computer. Well, *almost*.

HP: "I don't wanna boot Linux."

Me: "You're gonna."

HP: "NOOT NOOT!"

Me: *disguises rEFInd as Windows*

HP: "Welcome back, old friend!"

rEFInd now lives in witness protection, dressed as Windows, and Fedora loads like nothing ever happened.

Thank you for joining me.

And remember: next time someone tells you "UEFI just works,"
NOOT NOOT.

And if there's anybody here from HP; I'd love a quick word. *Outside!*