“Kernel command line to configure userspace” Considered Harmful

…or: “nice hornet nest you got there, be a shame if somebody kicked it”

Luca Boccassi
Linux Systems Group
Microsoft

Where we are and how did we get here

- In the beginning, there was the BIOS, and all was well
  - Security model: trust me bro
- In the 2000s, UEFI was created as successor
  - Having an actual security model doesn't mean being infallible! Things go wrong, bugs happen and there are processes to deal with it
  - Chain of trust starts in hardware, e.g.: Intel Boot Guard
  - Bootloader(s) and kernel signed by certificate in firmware -> SecureBoot
- In the 2010s, thanks to the work of many people, Linux joined the SecureBoot party with the Shim + Grub2 + Kernel Lockdown LSM stack
  - Generalist distros started first (SecureLevel patchset), upstream kernel followed later
  - Focused on protecting the integrity of the system before ExitBootServices(), and the firmware
  - Lockdown tries to enforce a security boundary between uid0 and ring0

Upcoming Changes to a Distro Near You

- UKIs (Unified Kernel Images) can extend the security model to ensure the initrd is trusted
  - It is part of the PE binary, signed and verified by SecureBoot
  - Status quo is that initrd is mutable and unverified
- IPE (Integrity Policy Enforcement) LSM allows writing a policy that ensures the integrity of the usr/rootfs is verified by the kernel, and allows enforcing code signing rules
- DDIs (Discoverable Disk Images) can protect payloads using signed dm-verity images, for extending the rootfs and for delivering/extending further discrete payloads
- When everything is added together, executed binaries in firmware, kernel and userspace, configuration in the initrd, vendor tree in usr/rootfs and other images, are all verified with root of trust in hardware
- …except for the kernel command line
  - Grub2 plain text file in the boot partition and in /etc/
  - systemd-boot's Type #1 BLS plain text files in ESP
  - Can be edited on-the-fly with serial console access

Kernel Command Line or Kitchen Sink?

- Document listing known kernel params is 7k lines long, and is defined as incomplete
  - That is quite a large surface!
- Parsed first in kernel EFI stub, before ExitBootServices()
- Used as if it was trusted/verified, but it's really not?
- Used by anything and everything
  - Can change/configure Linux Security Modules -> disable SELinux, IPE, etc.
  - Not just components in the kernel, but userspace too!
- Parsed by anything and everything
  - /proc/cmdline available by default for every userspace process
- In Confidential Computing, virtual console is outside TCB

Can we do something about this? Should we?

- Complex problem, due to the convenience of adding everything to the same place, and editing from the boot menu, it's the main way to customize a Linux boot, not just in the kernel but userspace too
  - E.g.: root filesystem selection, network configuration
- …but there is no real reason why **userspace** has to be configured via the **kernel** command line, other than history and convenience
  - Which, to be fair, are pretty strong reasons
- systemd-boot when loading Boot Loader Specification Type 2 payloads (UKIs) with SecureBoot enabled does not allow untrusted/unverified kernel command line changes
- All the flexibility and convenience is gone. How can we get some of it back?

Rootfs autodiscovery

- A major use case is configuring which rootfs to use when transitioning from the initrd to the OS
- root=/dev/sda1 or root=b993a5ba-9aff-4b2d-a9cd-33397975937f or other variations in OSTree-like environments
- The Discoverable Partitions Specification can cover quite well the case where the rootfs is fixed
- systemd-gpt-auto-generator finds the rootfs looking for the right GUID on the disk that was used to boot the system, using an EFI variable set by bootloaders following the Boot Loader Interface
    - Both systemd-boot and Grub2 support this!

UKIs

- When building the UKI, the command line string is embedded in the PE file as a section, so that it is covered by the signature and verified by UEFI SecureBoot
- ukify makes building UKIs with custom command line very simple
- Alternatetively/in addition a BootConfig file can be added to the initrd defining further command line entries
- But it is a fixed, single entry
- Future plan: multiple alternative entries in UKI, select one at boot menu
  - Common use case: want to enable debug logs
- Very high on TODO list, but not done yet

systemd-stub Addons

- systemd-stub implements support for “Addons” since v254
- Signed extension of the kernel command line, verified by SecureBoot
- Can be built by easily by ukify
- PE binary that can be dropped in the ESP, in /loader/addons/ for globally applicable ones, and in EFI/linux/uki.efi.extra.d/ for UKI-specific ones
- While the UKI is distributed by the OS vendor, addons are meant for platform/infrastructure owners, for platform-specific configuration
  - Common use case: platform-specific kdump settings, e.g.: crashkernel=256M
- TODO: select which addons to use from the boot menu

Extension Images

- Extension Images are signed dm-verity read-only images
- Stacked on /usr/ (sysext) or /etc/ (confext) to respectively extend vendor tree or configuration
- Can be dropped in the ESP as EFI/linux/uki.efi.extra.d/img.raw to extend the initrd, or in the initrd/rootfs under /var/lib/ (or other locations) to extend the rootfs
- Meant for both OS vendor (optional stack, e.g.: exotic drivers) and platform owner (local changes)
- Can be used to configure any userspace component given systemd loads them in the initrd already, including rootfs configuration
    - Common use case: complex storage components

Credentials

- systemd credentials are key/value pairs
- Scoped to single service, opt-in by key, not globally visible by default
- Can be provided via SMBIOS type 11 strings by hypervisor/VMM
  - Both QEMU and Cloud-Hypervisor support this!
- Can be passed through to containers down the chain
- Can be encrypted and authenticated using the TPM2
- Can be dropped in the ESP, globally as /loader/credentials/*.cred or per-UKI as EFI/Linux/uki.efi.extra.d/*.cred
  - (currently all picked up, on the TODO list: select a subset via boot menu)
- Can be synthesized from systemd-boot menu (implementation in progress, need to sort out measurement story)

Are we there yet?

- With gpt-auto, UKIs, addons, extension images and credentials, do we have all sensible and supportable use cases covered?
- As reviewers and maintainers of userspace components, should we start requesting that new additions to the kernel command line for our programs should instead (or also) use one of the aforementioned mechanisms?
- UEFI SecureBoot 2.0 is on the (distant) horizon. Should it say something about untrusted sources manipulating the kernel behaviour, at least before ExitBootServices()?
- Is this a fool’s errand?
  - probably

Thanks!

- Questions?