

NEW DESIGN FOR INTRDS

Zbigniew Jędrzejewski-Szmek



zbyszek@in.waw.pl



LPC 2022, 12.09.2022

(instead of) Intro

see Lennart's

“TOWARDS SECURE UNIFIED KERNEL IMAGES
FOR GENERIC LINUX DISTRIBUTIONS AND EVERYONE ELSE”

- “let's pre-build initrds in vendor build system”
- “System extension images are GPT disk images, implementing the Discoverable Partitions Specification”
- “Signed as one for SecureBoot”
- “systemd's «service credentials» are a concept for passing identity information, certificates, key material, passwords, and similar to services”

- also: reproducible builds
- also: a simpler system

Current approach to initrds

goals:

- speed → event-driven logic
- speed → size minimization
- flexibility, versability, end-user choice
- local configuration embedded in the initrd

results:

- local builds
- custom logic (e.g. dracut's initqueue)
- custom tools (e.g. scripts to bring up lvm, dracut modules)
- a unique execution environment
- the packaging layer is duplicated
- complexity (in particular when dracut is used with systemd)
- lots of CPU cycles burnt during each kernel update

What does the kernel say?

(the short answer: it doesn't care)

the long answer: the initrd is just an in-memory file system
`/init` is started instead of `/sbin/init`

New goals

- reuse distro packaging
 - use systemd in the initrd
 - use normal services
 - standard userspace environment
 - reasonable size
-
- build reproducible initrd images
 - build initrd images on vendor systems
 - sign the kernel + initrd
 - build a set of System Extension images for the initrd
 - sign those too
-
- maintainers of user-space packages handle “initrd bugs”

What are System Extensions good for?

Reminder: the initrd is a compressed cpio archive

a sysext is a GPT image with three partitions:
the filesystem (e.g. compressed squashfs), dm-verity
for the filesystem, signature for the verity data

- a network configuration daemon + sshd
- iSCSI / nfs / RAIDs / clevis / storage
- the full graphical stack: a11y! i18n!
- the full sound stack: a11y!
- hardware enablement, incl. bluetooth
- (suggestions welcome)

mkosi — introduction

<https://github.com/systemd/mkosi>

- A program that builds “images” from packages (and sources)
- Support for GPT, verity, and signature
- Also cpio images (initrd/initramfs)
- Uses dnf / apt / pacman / zypper / ... as backend
→ supports Fedora / CentOS / RHEL / Mageia /
OpenMandriva / Debian / Ubuntu / Arch / OpenSUSE /
Gentoo (sic!)
- `mkosi.skeleton/`, `mkosi.extra/`,
`mkosi.build`, `mkosi.postinst`, `mkosi.finalize`
- Support for incremental builds
- Writes “manifests” of installed packages
- We are working on build reproducibility

mkosi-initrd

<https://github.com/systemd/mkosi-initrd>

- mostly a series of a config files for `mkosi`
- list of packages for the “basic” `initrd`
- `mkosi.finalize` to set `/etc/initrd-release`
- set of configs for `sysexts`

- also
`/usr/lib/kernel/install.d/50-mkosi-initrd.install`

- “systemd credentials” will be used for configuration and local assets

Benefits

- **less** things
- we use package dependency resolution mechanism
- we let rpm/deb/pacman handle 90% of the installation
- we don't pull files from the host
- images can be reproducible
- images are the same for everyone
- images can be easily signed
- systemd does the heavy lifting in the initrd
- bash helpers → compiled programs
- developers don't need to learn another system
(initrd is like a normal system,
just on an fs not backed by a disk)
- clear ownership of bugs
- initrd infrastructure can be shared between distros

Links

<https://github.com/systemd/mkosi>

<https://github.com/systemd/mkosi-initrd>

<https://www.freedesktop.org/software/systemd/man/systemd-sysext.html>

<https://gitlab.com/cryptsetup/cryptsetup/-/wikis/DMVerity>

<https://www.kernel.org/doc/html/latest/admin-guide/device-mapper/verity.html>

<https://www.kernel.org/doc/html/latest/filesystems/overlayfs.html>

These slides:

<https://github.com/keszybz/mkosi-initrd-talk/raw/main/lpc2022-new-design-for-initrds.pdf>

QUESTIONS? / EOF