Building initrds in a new way

Zbigniew Jędrzejewski-Szmek

zbyszek@in.waw.pl

FOSDEM 2023, 4.2.2023

(instead of) Intro

(instead of) Intro

see Lennart's
"Image-Based Linux and TPMs
Measured Boot, Protecting Secrets and you"

- SecureBoot signing used to protect code
- "PCR measurements", "boot phases"
- systemd's «credentials» are a mechanism to pass identity information, certificates, key material, passwords, and similar
- "credentials" locked to PCR state
- systemd's «system extensions» are a mechanism to dynamically extend the root fs

Current approach to initrds

Current approach to initrds

- local builds pulling in files from host fs, `ldd` to resolve dependencies

Current approach to initrds

- local builds pulling in files from host fs, `ldd` to resolve dependencies
- the packaging layer is duplicated

Current approach to initrds

- local builds pulling in files from host fs,
  `ldd` to resolve dependencies
- the packaging layer is duplicated
- lots of CPU cycles burnt during each kernel update

Current approach to initrds

- local builds pulling in files from host fs,
  `ldd` to resolve dependencies
- the packaging layer is duplicated
- lots of CPU cycles burnt during each kernel update

- at runtime: custom logic (e.g. dracut's initqueue)

Current approach to initrds

- local builds pulling in files from host fs,
  `ldd` to resolve dependencies
- the packaging layer is duplicated
- lots of CPU cycles burnt during each kernel update

- at runtime: custom logic (e.g. dracut's initqueue)
- custom tools (e.g. scripts to bring up LVM, dracut modules)

Current approach to initrds

- local builds pulling in files from host fs,
  `ldd` to resolve dependencies
- the packaging layer is duplicated
- lots of CPU cycles burnt during each kernel update

- at runtime: custom logic (e.g. dracut's initqueue)
- custom tools (e.g. scripts to bring up LVM, dracut modules)
- different execution environment

Current approach to initrds

- local builds pulling in files from host fs,
  `ldd` to resolve dependencies
- the packaging layer is duplicated
- lots of CPU cycles burnt during each kernel update

- at runtime: custom logic (e.g. dracut's initqueue)
- custom tools (e.g. scripts to bring up LVM, dracut modules)
- different execution environment
- complexity (in particular when dracut is used with systemd)

Current approach to initrds

- local builds pulling in files from host fs,
  `ldd` to resolve dependencies
- the packaging layer is duplicated
- lots of CPU cycles burnt during each kernel update

- at runtime: custom logic (e.g. dracut's initqueue)
- custom tools (e.g. scripts to bring up LVM, dracut modules)
- different execution environment
- complexity (in particular when dracut is used with systemd)

- very little sharing of initrd logic between distros

Consequences of signing and measurement

Consequences of signing and measurement

- signing of the kernel but not the initrd is a waste of CPU cycles

Consequences of signing and measurement

- signing of the kernel but not the initrd is a waste of CPU cycles
- end-users want kernel+initrd signed by the distro

Consequences of signing and measurement

- signing of the kernel but not the initrd is a waste of CPU cycles
- end-users want kernel+initrd signed by the distro
- the initrd must be built by the distro

Consequences of signing and measurement

- signing of the kernel but not the initrd is a waste of CPU cycles
- end-users want kernel+initrd signed by the distro
- the initrd must be built by the distro
- flexibility & ability to inject local modifications — not useful

Consequences of signing and measurement

- signing of the kernel but not the initrd is a waste of CPU cycles
- end-users want kernel+initrd signed by the distro
- the initrd must be built by the distro
- flexibility & ability to inject local modifications — not useful

- if we are building in a package builder, let's build directly from distro packages
  (we *could* build from files in the fs, but why?)

Consequences of centralized builds

If we use pre-built images, two ways to deliver differentiated code

Consequences of centralized builds

If we use pre-built images, two ways to deliver differentiated code

1. initrd variants

Consequences of centralized builds

If we use pre-built images, two ways to deliver differentiated code

1. initrd variants
2. systemd-sysexts

Consequences of centralized builds, ctd.

If we use UKIs, we need a mechanism to replace config in the initrd

Consequences of centralized builds, ctd.

If we use UKIs, we need a mechanism to replace config in the initrd

1. automatic discovery (Discoverable Partitions Spec)

Consequences of centralized builds, ctd.

If we use UKIs, we need a mechanism to replace config in the initrd

1. automatic discovery (Discoverable Partitions Spec)
2. credentials for configuration

Consequences of centralized builds, ctd.

If we use UKIs, we need a mechanism to replace config in the initrd

1. automatic discovery (Discoverable Partitions Spec)
2. credentials for configuration

Easy building of sysexts depends build reproducibility of the initrd

mkosi — introduction

`https://github.com/systemd/mkosi`

mkosi — introduction

`https://github.com/systemd/mkosi`

- A program that builds “images” from packages (and sources)

mkosi — introduction

`https://github.com/systemd/mkosi`

- A program that builds "images" from packages (and sources)
- Support for GPT, verity, and signatures → sysexts

mkosi — introduction

`https://github.com/systemd/mkosi`

- A program that builds "images" from packages (and sources)
- Support for GPT, verity, and signatures → sysexts
- Also archives (`cpio`) → initrd/initramfs

mkosi-initrd

`https://github.com/systemd/mkosi-initrd`

mkosi-initrd

`https://github.com/systemd/mkosi-initrd`

- mostly a series of a config files for `mkosi`

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-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`

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

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`

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`

- hopefully coming soon to a Fedora install near you
  `https://fedoraproject.org/wiki/Changes/mkosi-initrd`

Results

Results

- We get fully functional initrds.

Results

- We get fully functional initrds.
- The initrds are **bigger**.

Results

- We get fully functional initrds.
- The initrds are **bigger**.
  Most of the difference is caused by kernel modules.

Results

- We get fully functional initrds.
- The initrds are **bigger**.
  Most of the difference is caused by kernel modules.
- Only some subset of installations is supported.

Benefits

- **less** things

Benefits

- **less** things
- we use package dependency resolution mechanism

Benefits

- **less** things
- we use package dependency resolution mechanism
- we let rpm/deb/pacman handle 90% of the installation

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

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

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

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

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

Objections?

Objections?

- systemd was already used in the initrd

Objections?

- systemd was already used in the initrd
- the first thing systemd does is to set up the environment

Objections?

- systemd was already used in the initrd
- the first thing systemd does is to set up the environment
- having tools that support running in a custom environment is hence not useful

Objections?

- systemd was already used in the initrd
- the first thing systemd does is to set up the environment
- having tools that support running in a custom environment is hence not useful
- after removing custom logic we don't need to add anything back

Objections?

- systemd was already used in the initrd
- the first thing systemd does is to set up the environment
- having tools that support running in a custom environment is hence not useful
- after removing custom logic we don't need to add anything back
- the ecosystem is moving away from scripts towards compiled daemons

Objections?

- systemd was already used in the initrd
- the first thing systemd does is to set up the environment
- having tools that support running in a custom environment is hence not useful
- after removing custom logic we don't need to add anything back
- the ecosystem is moving away from scripts towards compiled daemons
- most of the code is in shared libraries, which are installed in full because of link dependencies

Objections?

- systemd was already used in the initrd
- the first thing systemd does is to set up the environment
- having tools that support running in a custom environment is hence not useful
- after removing custom logic we don't need to add anything back
- the ecosystem is moving away from scripts towards compiled daemons
- most of the code is in shared libraries, which are installed in full because of link dependencies
- error handling, timeouts, retries, localized messages, event-driven logic, netlink, D-bus, all are much easier with "real" code

Progress over the last few months

- `mkosi-initrd` has a growing test suite (booting different storage types)
- `mkosi` supports builds as an unprivileged user
- `systemd` is getting new credential features
- `systemd` has new `ukify` helper to builds UKIs
- Fedora 38 Change accepted for Unified Kernel Images for VMs
- New kernel package split in Fedora (`kernel-modules-core` finally)
- GRUB2 might get support for UKIs (`https://github.com/osteffenrh/grub2`)
- Fedora 39 Change proposal for `mkosi-initrd`

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/fosdem2023-buildling-initrds-in-a-new-way.pdf

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/fosdem2023-buildling-initrds-in-a-new-way.pdf

QUESTIONS? / EOF