# Unprivileged CRIU

Younes Manton, Linux Plumbers' Conf 2022

# Outline

- Unprivileged CRIU: Dump & restore without root
  - Original patch set
  - Current patch set
  - Unresolved questions
- Use-case: Improve JVM start-up time
  - Motivation
  - Challenges

# Unprivileged CRIU: Dump & restore without root

- Allows for limited checkpoint & restore without root
- Set `CAP_CHECKPOINT_RESTORE` on criu executable
- Pass `--unprivileged` to both dump and restore sub-commands
- May or may not work, some things cannot (currently) be done without root
- My approach: check if it works as root first, then try unprivileged

# Unprivileged CRIU: Original patch set

- PR #1155 on GitHub
- Opened in mid 2020
- From Adrian Reber, based on his and Nicolas Viennot's earlier work
- Needs the (not so) recently added CAP_CHECKPOINT_RESTORE capability
  - Works with CAP_SYS_ADMIN as a substitute
- Probably needs others, depending on program & environment in question
  - CAP_SYS_PTRACE (or set ptrace_scope to 0)
  - CAP_NET_ADMIN
  - CAP_SYS_CHROOT
  - CAP_SYS_RESOURCE
  - CAP_SETUID

# Unprivileged CRIU: Current patch set

- PR #1930 on GitHub
- Opened in mid 2022
- From myself, based on #1155
- Rebased and modified to accommodate recent changes in criu-dev
- More or less a continuation of Adrian’s work
- Added RPC / libcriu support
- `kerndat` file now written to `XDG_RUNTIME_DIR` if set

# Unprivileged CRIU: Unresolved questions

- `fchroot()`'s behaviour without CAP_SYS_CHROOT
- Default non-root behaviour, require `--unprivileged` or assume?
- Avoid known limitations v.s. deterministically fail
- Options to skip certain ops? `--skip-x`, `--skip-y`
- Options to ignore certain caps? `--no-cap-x`, `--no-cap-y`
- `has_root_cg_set` & `dump_task_cgroup()` behaviour
- Should options related to unprivileged mode be not guaranteed to be stable until a later date?

# Use-case: Improve JVM start-up time

# Use-case: Improve JVM start-up time - Motivation

- Start-up time is a priority in the cloud/microservices world
- JVM start-up is slow, provides a lot of services, can only be optimized so much
  - An “empty main” program is 100x slower
- JIT compilation conflicts with fast start-up
- AOT helps, but not enough (and code is inferior)
- JVM is big, complicated; paradigm shifts are difficult
- Checkpoint/restore offers a way to solve many problems

# Use-case: Improve JVM start-up time - Challenges

- Restoring a process is not like starting a process from scratch, JVM and/or programs need to be C/R-aware
- Some user expectations are inherently broken by C/R, need attention
- Unprivileged CRIU addresses one such expectation: users don’t expect to run JVM as root
- Users expect to to be able to pass command line and environment; restored processes don’t see new cmdline, env
- C/R is less portable; process migration users expect/accept it, start-up users may not
- CAP_CHECKPOINT_RESTORE not yet everywhere
  - Docker!

# Backup: OpenJ9’s CRIU Usage & Awareness

- Link with `libcriu`
- Expose a `dump()` et. al. Java API (maps more or less to CRIU’s API)
- Prepare JVM for dump
- Call `libcriu` to configure and perform dump
- Invoking restore is left to application to handle
- On restore, do necessary magic to keep things working

# Backup: Java Application's CRIU Usage & Awareness

- Provide a script to for users to start your application
- Try to restore if possible by invoking `criu restore …`
- Fall back to conventional start-up otherwise
- Prior to calling our `dump()` do what you need to make yourself restorable
- After our `dump()` returns, do what you need to keep things working

Thank You!