Google

Securely migrating untrusted workloads with CRIU

Linux Plumbers Conference 2018

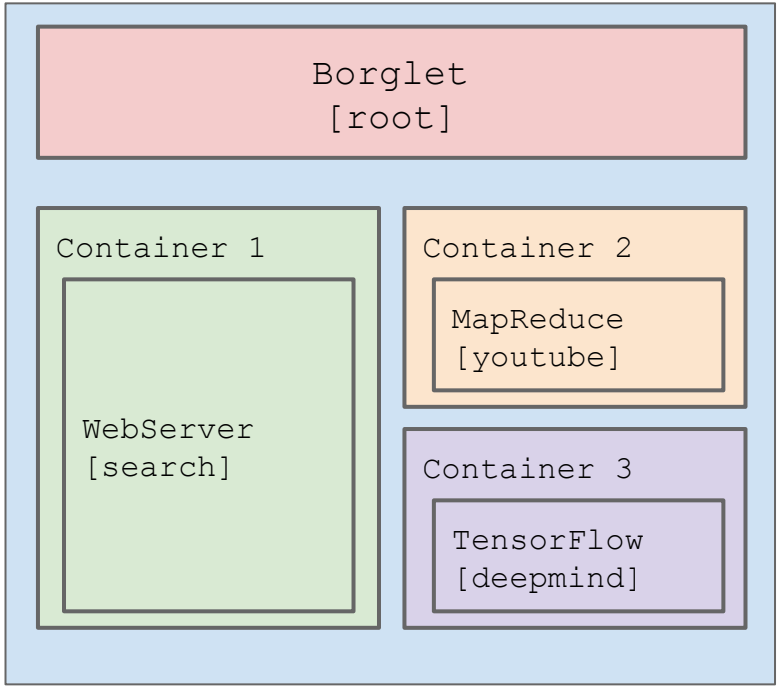
Radoslaw Burny
rburny@google.com

2018-11-15

Borg isolation

Borg runs multiple tasks on the same machine, managed by a “Borglet” daemon.

- tasks are isolated by containers
  - cgroups + namespaces + chroot
- tasks are considered untrusted
  - must be isolated from each other
- tasks are not privileged
  - i.e. no Linux capabilities



Where does CRIU fit in the picture?

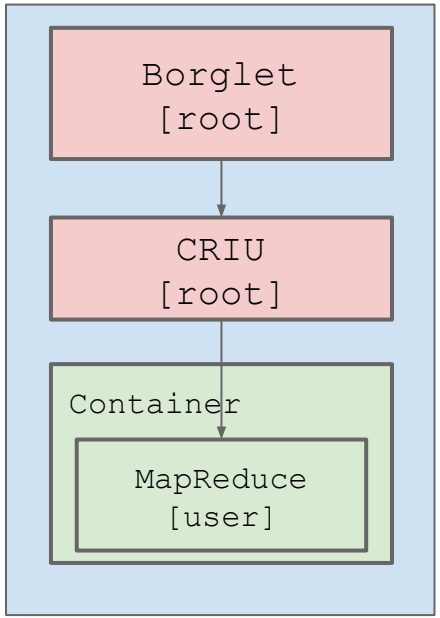
Running CRIU

CRIU performs complex work on behalf of tasks...

- uses breadth of kernel interfaces
- requires elevated capabilities

It’s easiest to run CRIU as root.

In theory, it’s safe - CRIU drops capabilities during restore, before returning control to the user code.



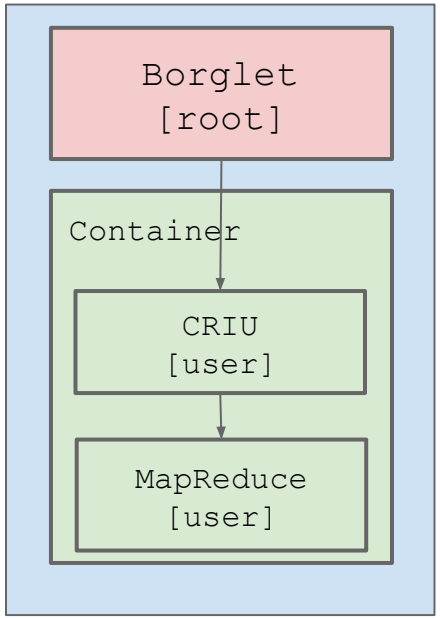
Running CRIU securely

CRIU performs complex work on behalf of tasks...

- a malicious task could exploit it
- … and gain its capabilities

We need to run CRIU as the task's user, with minimal caps.

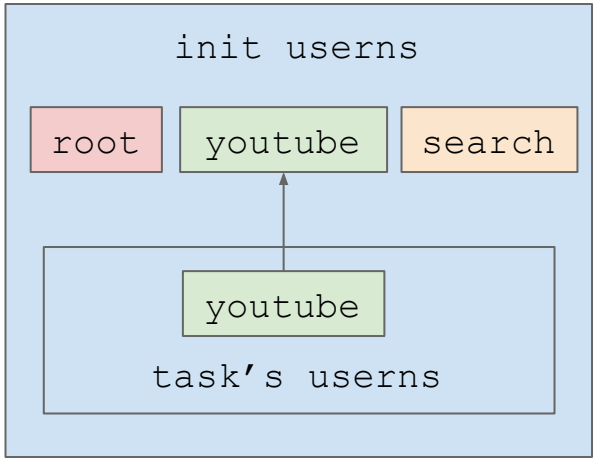
Bonus: non-privileged apps can also use CRIU
(example: build system restores prewarmed Java compiler).



Step 1- user namespace without root

Run tasks (and CRIU) in userns without root mapping.

- capabilities in userns don't map to global ns
- if user exploits a bug and gains control of userns
  -> they still have no access to global root



Seems like we were not the first ones to try it:

```
Subject  [PATCH] prctl: Allow local CAP_SYS_ADMIN changing exe_file
From     Kirill Tkhai <>
Date     Fri, 12 May 2017 17:33:36 +0300

During checkpointing and restore of userspace tasks
we bumped into the situation, that it's not possible
to restore the tasks, which user namespace does not
have uid 0 or gid 0 mapped.
```

Step 2 - capability reduction

Run CRIU with task's user's credentials. Minimize the number of additional Linux capabilities by avoiding privileged operations:

- don't migrate cgroups & namespaces (Borglet recreates them)
- check if the setting is already at a desired value, avoid redoing it
  - `chroot, setgroups, chown, /proc/self/loginuid, ...`
- disable privileged parts of socket migration code
  - we currently break & re-establish network connections anyway
  - will eventually need to revisit this to allow non-disruptive migration

Capability reduction - results

We’re down to two functionalities requiring a capability.
Both occur on restore and require local CAP_SYS_ADMIN:

1. writing to `/sys/kernel/ns_last_pid`
   - workaround: delegate to privileged helper process
2. changing `/proc/$PID/exe` via `prctl(PR_SET_MM, PR_SET_MM_MAP, …)`
   - no known workaround

Both interfaces originated from CRIU project.
Are the strict capability requirements really necessary?

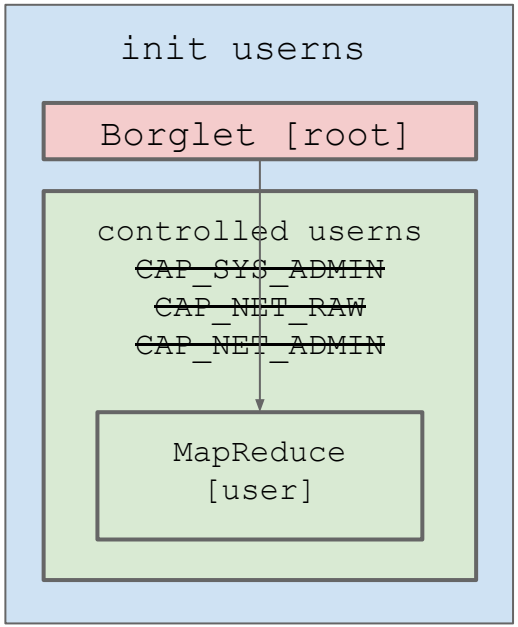
Controlled user namespaces

User namespaces can be used to exploit bugs:

- create user namespace, get all caps, exploit!
- “solution”: limit the ability to create userns

Mahesh Bandewar proposed “controlled” userns:

- only whitelisted capabilities can be gained
- children namespaces also become “controlled”
- thus, a process running in a “controlled” userns can never gain “dangerous” capabilities



Capability reduction is necessary to run in a “controlled” userns.

Thank you!

Our questions:

- is the community interested in running CRIU unprivileged?
- can we reduce cap requirements for `ns_last_pid` and `PR_SET_MM_MAP`?