Minijail:
Sandboxing software running on Linux kernels

Jorge Lucangeli Obes
jorgelo@google.com

```
$ ps -eo pid,user,comm
```

```
   1 root     /sbin/init
   2 root     [kthreadd]
 ...
1468 message+ dbus-daemon --system --fork
1749 root     /usr/sbin/bluetoothd
1948 root     rsyslogd
2063 root     /usr/sbin/ModemManager
4159 root     NetworkManager
4175 tss      /usr/sbin/tcsd
4571 root     /sbin/wpa_supplicant
4586 root     /sbin/dhclient
 ...
```

BUT WHY?

Misaligned incentives

Admins don’t know what permissions the software needs.

Devs don’t know where their software runs.

And when they try…

```
void switchUser() {
    …
    prctl(PR_SET_KEEPCAPS, 1, 0, 0, 0);
    setuid(user);

    (15+ lines setting up |header|, |data|)
    if (capset(&header, &data[0]) == -1) {
```

Let’s stop reinventing the wheel!

```
void switchUser() {
    …
    struct minijail *j = minijail_new();
    minijail_change_uid(j, <uid>);
    minijail_use_caps(j,
                      CAP_MASK_LONG(CAP_BLOCK_SUSPEND) |
                      CAP_MASK_LONG(CAP_NET_ADMIN) |
                      CAP_MASK_LONG(CAP_NET_RAW));
    minijail_enter(j);
```

And also…

Piñata Party
4,705
x3
1200
x1

Minijail: Android, Chrome OS, and Brillo's ~~sandboxing~~ containment helper.

UID, GID, supplementary groups

```
# id

uid=0(root) gid=0(root) groups=0(root),125(pkcs11)



# minijail0 -u jorgelo -g eng -G -- /usr/bin/id

uid=72178(jorgelo) gid=5000(eng) groups=5000(eng), 499(google),
5001(guest), ...
```

Not impressed.

Capabilities

```
# minijail0 -u jorgelo -c 3000 -- /bin/cat /proc/self/status

Name:   cat
...

Uid:    72178   72178   72178   72178
Gid:    0   0   0   0
CapInh: 0000000000003000
CapPrm: 0000000000003000
CapEff: 0000000000003000
CapBnd: 0000000000003000

...
```

```
      1 root     /sbin/init
      2 root     [kthreadd]
    ...
   1468 message+ dbus-daemon --system --fork
   1749 bluetoo+ /usr/libexec/bluetooth/bluetoothd
   1948 syslog   rsyslogd
   2063 modem    /usr/sbin/ModemManager
   4571 wpa      /usr/sbin/wpa_supplicant
   4586 dhcp     /sbin/dhcpcd
    ...
```

What about kernel bugs?

Seccomp

```
# minijail0 -u jorgelo -n -S test/cat.policy -- \
    /bin/cat /proc/self/status
Uid:    72178  72178  72178  72178

...
CapInh:    0000000000000000
CapPrm:    0000000000000000
CapEff:    0000000000000000
CapBnd:    0000000000000000
Seccomp:   2
```

Seccomp policy

```
read: 1
write: 1
rt_sigreturn: 1
exit: 1

open: 1
close: 1
fstat: 1
mmap: 1
fadvise64: 1
```

Seccomp

```
# minijail0 -u jorgelo -n -S test/cat.policy -- \
    /bin/cat /proc/self/status
Uid:    72178  72178  72178  72178
...
```

LD_PRELOAD

```
static int (*real_main) (int, char **, char **);
static void *libc_handle;


int API __libc_start_main(...) {
    ...
    libc_handle = dlopen("libc.so.6", RTLD_NOW);
    ...
```

LD_PRELOAD

```
static int fake_main(int argc, char **argv, char **envp) {
    minijail_preenter(j);
    minijail_enter(j);
    minijail_destroy(j);
    dlclose(libc_handle);
    return real_main(argc, argv, envp);
}
```

Capability inheritance over execve(2)

```
P’(permitted) = (P(inheritable) & F(inheritable)) |
                (F(permitted) & cap_bset)

P’(effective) = F(effective) ? P’(permitted) : 0

P’(inheritable) = P(inheritable)
```

Without file capabilities (F(*) = 0), P’(permitted), **P(effective)** will be 0.

Ambient capabilities

```
P’(amb)  = (file caps or setuid or setgid ? 0 : P’(amb))

P’(perm) = (cap_bset & F(perm)) | (P(inh) & F(inh)) | P’(amb)

P’(inh)  = P(inh)

P’(eff)  = (F(eff) ? P’(perm) : P’(amb))

cap_bset is unchanged.
```

If you are non-root but you have a capability, you can add it to P(ambient).
If you do so, your children get that capability in P(ambient), P(permitted), and P(effective).

What if restricting is not feasible?

Namespaces

PID namespaces

```
# ./minijail0 -p -- /bin/ps
   PID TTY          TIME CMD
     1 pts/27   00:00:00 minijail0
     2 pts/27   00:00:00 ps
```

PID/mount namespaces

```
# ./minijail0 -p -- /bin/ls -l /proc
total 0
dr-xr-xr-x  9  root root  0 Aug  9 10:38 1
dr-xr-xr-x  9  root root  0 Aug  9 10:38 2
dr-xr-xr-x  4  root root  0 Aug  9 10:38 acpi

...
```

PID/mount namespaces

```
# readlink /proc/self/ns/mnt
mnt:[4026531840]

# ./minijail0 -v -- /bin/readlink
/proc/self/ns/mnt
mnt:[4026532712]
```

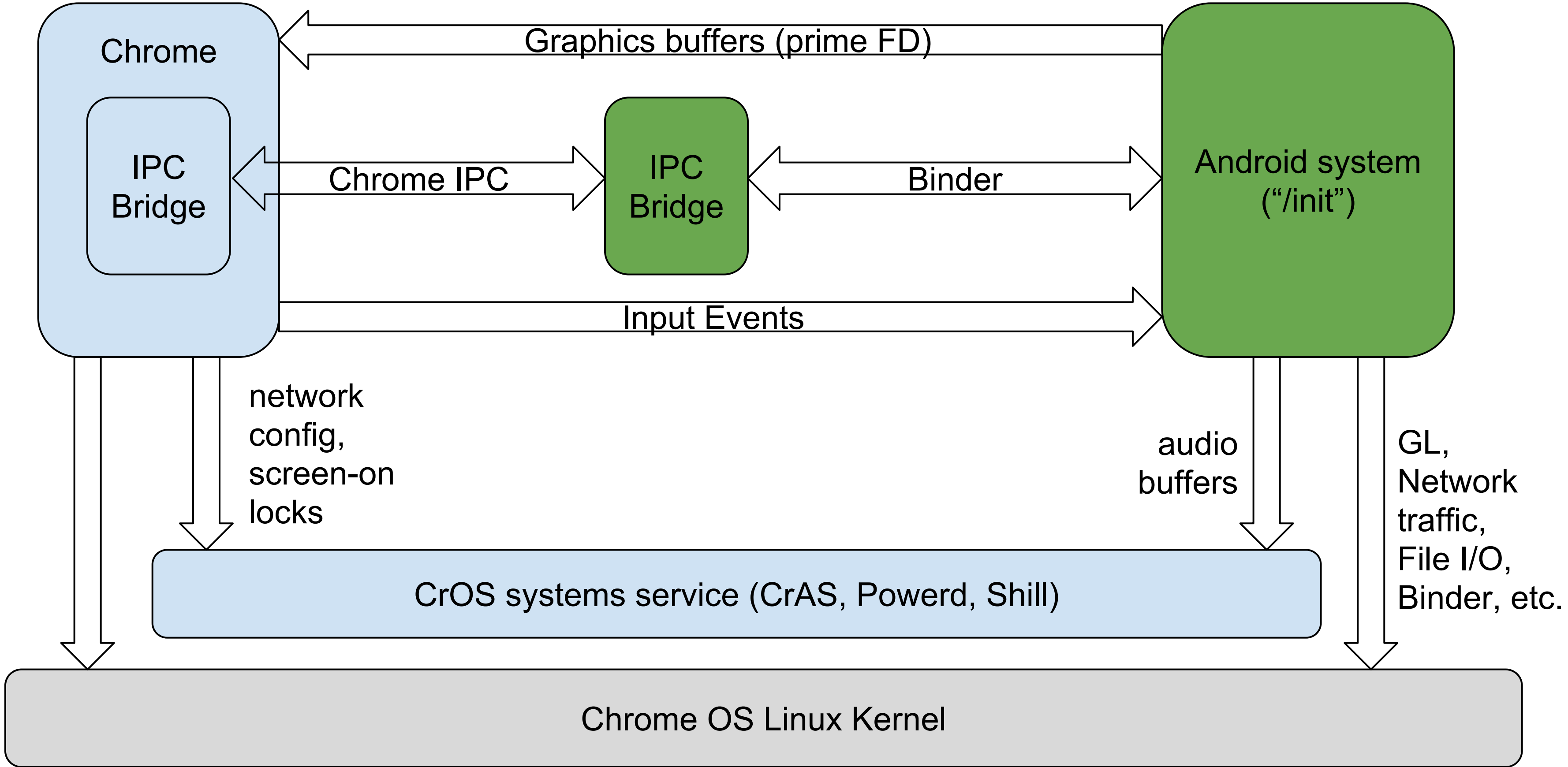
User namespaces

```
$ id -u
72178
$ ./minijail0 -U -m -- /usr/bin/id -u
0
```

And we don’t need to run this as `root`!

Which brings us to...

Android container


Chrome
IPC Bridge
Graphics buffers (prime FD)
Chrome IPC
IPC Bridge
Binder
Android system ("/init")
Input Events
network config, screen-on locks
audio buffers
GL, Network traffic, File I/O, Binder, etc.
CrOS systems service (CrAS, Powerd, Shill)
Chrome OS Linux Kernel

Android container

```
minijail_namespace_ipc(c->jail);
minijail_namespace_vfs(c->jail);
minijail_namespace_net(c->jail);
minijail_namespace_pids(c->jail);
minijail_namespace_user(c->jail);

rc = minijail_uidmap(c->jail, config->uid_map);
rc = minijail_gidmap(c->jail, config->gid_map);

rc = minijail_enter_pivot_root(c->jail, c->runfsroot);
rc = minijail_add_to_cgroup(c->jail...
rc = minijail_run_pid_pipes_no_preload(c->jail...
```

Acknowledgments

Will Drewry wrote the initial version of Minijail.

Elly Jones rewrote Minijail in C and implemented the preloading mechanism.

Dylan Reid wrote a big chunk of the container/namespace support.

Chrome OS team at Google contributed container-related functionality.

Lee Campbell wrote the ELF parsing code and the initial support for static binaries.

Kees Cook reviewed a lot of code.

Questions?

For more info:

go/minijail

g/minijail-users

g/minijail-dev

Back-up slides

Network namespaces

```
# ifconfig
em1       Link encap:Ethernet  HWaddr ...
          inet addr:172.31.196.11  ...

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host

# ./minijail0 -e -- /sbin/ifconfig
#
```