$ snappy-debug

demystify & extend snapd

A snap is not a container. It is a normal process wearing four sets of kernel restraints at once.

Soumyadeep Ghosh

UbuCon Asia 2026

08 Aug 2026

ABOUT ME

Soumyadeep Ghosh

now	Canonical, Store team, SaaS Engineering
community	Ubuntu Member since 2024, ,member of Snapcrafters since 2023
since 2022	contributing to snapd, snapcraft and App Center

AGENDA

Four “on”, one process

1. What confinement actually is
2. Debugging denials, with snappy-debug
3. Extending snapd with a new interface
4. Habits for a better manifest

What confinement is

CONFINEMENT

A snap is not a container.

It is a normal Linux process wearing four sets of kernel restraints at once.

AppArmor	files, exec, mounts, IPC, signals, network
seccomp	an allowlist of syscalls, widened by interfaces
device cgroup	which devices this command may touch
DAC perms	owner, group and ACLs, still on top

snapcraft.io/docs/explanation/security/security-policies

CONFINEMENT

What a denial looks like

journalctl

```
❯ journalctl -k | grep -E 'DENIED|SECCOMP'

# AppArmor  ->  EACCES, and it is logged
Aug 04 02:39:43 soumyadeep-gf63thin11uc kernel: audit: type=1400 audit(1785791383.054:237338):
apparmor="DENIED" operation="open" class="file" profile="snap.firefox.firefox" name="/mnt/snappy-
demo/notes.txt" pid=138653 comm="cat" requested_mask="r" denied_mask="r" fsuid=1000 ouid=0

# seccomp   ->  EPERM, logged (just a syscall number)
Aug 04 12:07:08 soumyadeep-gf63thin11uc kernel: audit: type=1326 audit(1785825428.528:956):
auid=1000 uid=1000 gid=1000 ses=4 subj=snap.firefox.firefox pid=15617 comm="python3"
exe="/usr/bin/python3.12" sig=0 arch=c000003e syscall=248 compat=0 ip=0x704455d8528d code=0x50000

# device cgroup / file perms: nothing prints. that is the point.
```

real journal, this machine . security-policies

Debugging denials

Reading it by hand is hard

noise	denials are buried among everything else in the journal
numbers	seccomp logs a syscall number, resolve it with scmp_sys_resolver
silence	device cgroup denials are never logged at all

wiki.ubuntu.com/SecurityTeam/.../SnappyConfinement/Cheatsheet

It reads the log and names the fix

watches the journal for policy violations

shows them in a human readable form

recommends the interface that grants it

install and run

```
$ sudo snap install snappy-debug
$ sudo snappy-debug

# start it before launching the snap,
# it only sees what happens next
```

snapcraft.io/docs/how-to-guides/snap-development/debug-snaps

DEMO, FIREFOX

A denial you can see

snappy-debug

```
❯ sudo snappy-debug
[sudo: authenticate] Password:
INFO: Following '/var/log/syslog'. If dropped messages, use:
INFO: journalctl --output=short --follow --all | snappy-debug
kernel.printk_ratelimit = 0

= AppArmor =
Time: 2026-08-04T03:40:23
Log: apparmor="DENIED" operation="open" class="file" profile="snap.firefox.firefox"
name="/mnt/snappy-demo/notes.txt" pid=11975 comm="FSBroker162800" requested_mask="r"
denied_mask="r" fsuid=1000 ouid=0
File: /mnt/snappy-demo/notes.txt (read)
Suggestions:
* adjust program to read from $SNAP, $SNAP_DATA, $SNAP_COMMON, $SNAP_USER_DATA or
$SNAP_USER_COMMON
* add 'removable-media' to 'plugs'
```

DEMO, NEWSFLASH, THE GOTCHA

A denial you can't see

Disconnect network-status. Silent denial.

newsflash

```
❯ snap disconnect newsflash:network-status && snap run newsflash
[2026-08-04T07:17:07Z INFO news_flash_gtk::app::imp] plugin id Some(PluginID("freshrss"))
[2026-08-04T07:17:08Z WARN news_flash_gtk::app::imp] Omitting startup sync due to no network
connectivity
[2026-08-04T07:17:08Z INFO news_flash_gtk::util] Networkmonitor connectivity: Local (available
false)

❯ snap connect newsflash:network-status && snap run newsflash
[2026-08-04T07:18:01Z INFO news_flash] Initialize ArticleScraper
[2026-08-04T07:18:01Z INFO news_flash_gtk::util] Networkmonitor connectivity: Full (available
true)
```

Sometimes, denials can be silent though, you just can’t catch it.

> snapcraft.io/docs/network-status-interface

Extending snapd

EXTEND SNAPD

An interface is just a Go file

snappy_demo_support.go

```
package builtin

const snappyDemoSupportSummary = `allows
    access to a demo file`

const snappyDemoSupportConnectedPlug
AppArmor = `
/root/example.txt r,
`
func init() {
    registerIface(&commonInterface{
        name: "snappy-demo-support",
        implicitOnCore: true,
        implicitOnClassic: true,
    })
}
```

`summary`	one line, what it grants
`snippet`	one narrow AppArmor path
`init()`	registers it with snapd

FULL WALKTHROUGH
gist.github.com/soumyaDghosh/53fada7ec9e3a8abcb42c33f642e2e71

EXTEND SNAPD

Build, replace, test

1. clone snapd, git switch --detach a release, not HEAD
2. copy removable_media.go, rename it throughout
3. register a commonInterface in init(), one narrow path
4. add a test beside it, go test ./interfaces/builtin
5. go build -o /tmp/build/snapd ./cmd/snapd
6. stop system snapd, run your build in the foreground
7. install the consumer snap, plug it, run: denied
8. snap connect, run again, it prints it works

gist.github.com/soumyaDghosh/53fada7ec9e3a8abcb42c33f642e2e71

EXTEND SNAPD

Getting it upstream

where	github.com/canonical/snapd, interfaces/builtin
tests	every interface needs unit tests and spread tests
review	the security team reviews it, so explain the threat model
scope	small, well scoped interfaces land fastest
forum	a proposal gets shaped there before the code does

github.com/canonical/snapd

Three myths

"snaps are bloated"	a badly built snap is. a good one bundles only what it needs.
"snaps start slowly"	a snap is a squashfs image. the LZO compression default fixed the first-start lag.
"classic snaps are unconfined"	traditional access by design, plus manual store review.

snapcraft.io/docs/snapcraft-yaml-reference

Writing better manifests

SEVEN HABITS

Think like a manifest, not a bash script.

A snap patched together like a shell hack is slow, heavy, and hard to trust.

1. Per-arch source and packages use grammar, not shell
2. Patch pkg-config gently, includedir as ${prefix}
3. No layouts, fix the build prefix instead
4. Strict over classic, extend snapd if needed
5. Set env with the environment field, not wrappers
6. Strip development libraries at prime

snapcraft.io/docs/reference/advanced-grammar

MANIFEST, PER ARCHITECTURE

Grammar, not shell branching

snapcraft.yaml

```
parts:
  app:
    source:
      - on amd64: https://example.com/app-amd64.tar.xz
      - on arm64: https://example.com/app-arm64.tar.xz
    stage-packages:
      - on amd64: [ libfoo-amd64 ]
      - on arm64: [ libfoo-arm64 ]
```

snapcraft.io/docs/reference/advanced-grammar

One rule for every .pc

libfoo.pc, patched per path

```
# invasive, breaks on the next lib
prefix=/usr
includedir=/usr/include
```

libfoo.pc, one rule for all

```
# non-invasive, free at runtime
prefix=/usr
includedir=${prefix}/include
```

freedesktop.org/wiki/Software/pkg-config

MANIFEST, NO LAYOUTS

Fix the prefix, skip the bind mount

snapcraft.yaml

```
parts:
  app:
    plugin: meson
    meson-parameters:
      - --prefix=/snap/myapp/current/usr   # build where it will run
    organize:
      # gotcha: files also land in etc/, share/, libexec...
      # organizing only usr silently drops them. move everything:
      snap/myapp/current: .                # the whole tree, not just usr
```

snapcraft.io/docs/reference/plugins/meson-plugin

MANIFEST, CONFINEMENT

Classic is not a fallback

A snap can almost always be strict. If snapd lacks the interface it needs, that is the cue to extend snapd, not to give up confinement.

snapcraft.yaml

```
confinement: strict        # not: classic
plugs:
  my-new-thing:            # add the interface upstream
    interface: my-new-interface
```

snapcraft.io/docs/reference/interfaces

MANIFEST, ENVIRONMENT

Snapcraft fields, not a launcher

snapcraft.yaml

```
environment:                # for every app in the snap
  GTK_USE_PORTAL: "1"
apps:
  myapp:
    environment:             # for this app only
      MYAPP_DATA: $SNAP/share
```

snapcraft.io/docs/reference/snapcraft-yaml/environment

MANIFEST, SLIM THE PAYLOAD

Drop what runtime never needs

snapcraft.yaml

```
parts:
  app:
    prime:
      - -usr/include            # headers
      - -usr/lib/**/*.a         # static libs
      - -usr/lib/**/pkgconfig   # build metadata
```

snapcraft.io/docs/reference/parts/lifecycle

Thank you. Questions?

Soumyadeep Ghosh

github.com/soumyaDghosh

UbuCon Asia 2026