FIVE STAGES OF

KUBERNETES SECURITY GRIEF

A dual-perspective journey through denial, anger, bargaining, depression, and acceptance -- what people actually say, and what's actually happening in the kernel.

ANN WALLACE + KAVI DAULA

KUBECON + CLOUDNATIVECON NORTH AMERICA // NOV 10 // SALT LAKE CITY

PRESS START

EVERYONE IN THIS ROOM
HAS BEEN THROUGH IT.
MAYBE YOU'RE IN IT
RIGHT NOW.

[RAISE YOUR HAND]

THE FRAMEWORK

TWO LENSES, FIVE STAGES



01

DENIAL

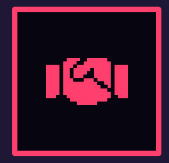
"Our namespace isolation is fine."



02

ANGER

"Why does everything touch the kernel?"



03

BARGAINING

"What if we just turn on Pod Security Admission?"



04

DEPRESSION

"The CVE dropped on a Friday."



05

ACCEPTANCE

"You can't bolt on isolation. You have to build it in."



WHAT PEOPLE SAY



WHAT THE KERNEL DOES

Every stage gets both lenses: the rationalization people reach for, and the kernel-level reality underneath it -- grounded in a real, public CVE.

01

DENIAL



“Our namespace isolation is fine.”

STAGE.01 -- DENIAL

“Our namespace isolation is fine.”



THE RATIONALIZATION

- We namespace everything by team, so tenant isolation is basically handled.
- It’s a shared cluster, sure, but the namespaces keep everyone in their lane.
- Compliance already signed off on the namespace boundary.

STAGE.01 -- DENIAL

"Our namespace isolation is fine."

THE TECHNICAL REALITY

CVE-2022-0492

Container calls unshare() -- creates its own user namespace

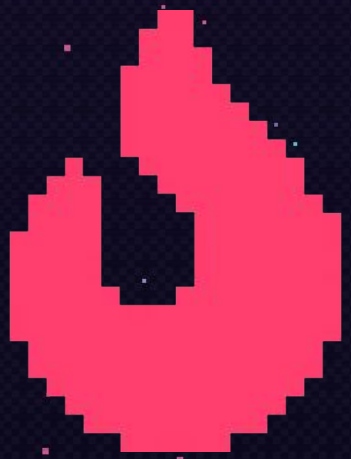
Gains CAP_SYS_ADMIN inside that namespace it just made

Kernel honors it as host-level -> writes cgroup release_agent

Nothing about this touches a namespace boundary -- it's a capability-check bug in cgroups, a completely different subsystem. "Our namespace isolation is fine" was never even the right sentence.

02

ANGER



“Why does everything touch the kernel?”

STAGE.02 -- ANGER

"Why does everything touch the kernel?"



SCOPE CREEP, AND IT FEELS UNFAIR

> A "container security" problem keeps turning into a kernel problem -- and every team hits the same wall: this wasn't supposed to be my kernel to worry about.

STAGE.02 -- ANGER

"Why does everything touch the kernel?"



IT'S ONE KERNEL, NOT ONE CONTAINER

CVE-2023-32233

Unprivileged user namespaces enabled -- a common host default

Process gets CAP_NET_ADMIN inside its own namespace

Reaches Netfilter nf_tables -> use-after-free -> kernel r/w

An ordinary packet-filtering bug becomes full root on the host -- for a workload that has nothing to do with networking, because it's all one kernel underneath. (Other unwatched surfaces: io_uring, the scheduler, the page cache, /proc.)

03

BARGAINING



“What if we just turn on Pod Security Admission?”

STAGE.03 -- BARGAINING

"What if we just turn on Pod Security Admission?"



THE NEGOTIATION PHASE

- What if we just turn on Pod Security Admission in restricted mode -- does that close it?
- Can we add an OPA policy for this instead of re-architecting?
- If we ban privileged containers outright, are we basically covered?

STAGE.03 -- BARGAINING

"What if we just turn on Pod Security Admission?"

POLICY CAN'T REACH THIS LAYER

CVE-2019-5736

Root in container replaces binary with #!/proc/self/exe

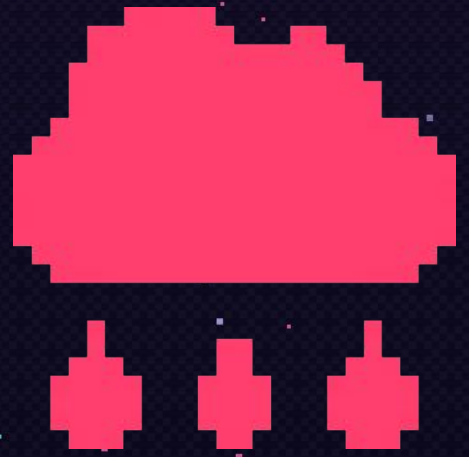
That path resolves to the host's runc binary

Overwrites host runc -> next container started runs as root

No runAsNonRoot, allowPrivilegeEscalation:false, PSP, or PSA restricted mode reaches this -- none of them touch the trusted runtime binary that mediates every container start.

04

DEPRESSION



“The CVE dropped on a Friday.”

STAGE.04 -- DEPRESSION

"The CVE dropped on a Friday."



THE HUMAN COST

- It's 4:45 on a Friday and legal wants a customer notification draft by Monday.
- We patched three environments and I still don't know if we caught all of them.
- How do we even find out what else we don't know here.

"The CVE dropped on a Friday."

AND THE ONES WITH NO CVE AT ALL

CVE-2024-21626

A Dockerfile WORKDIR triggers chdir into the container dir

Points chdir at /proc/self/fd/<n> -- a still-open host fd

Container reaches the host filesystem with runtime-level root

"Leaky Vessels" -- four coordinated CVEs across runc, BuildKit, and Docker; a real patch scramble. The scarier category is the one with no advisory at all: gaps that were never a boundary to begin with, so there's nothing to point to. [swap in your own example here]

05

ACCEPTANCE



“You can't bolt on isolation. You have to build it in.”

THE TECHNICAL SIDE LEADS THIS ONE -- THE INVERSION IS INTENTIONAL

"You can't bolt on isolation. You have to build it in."



WHAT "BUILT IN" ACTUALLY MEANS

Namespaces, cgroups, seccomp, PSA -- independently-imperfect

Each patchable; several reachable from "underneath" the last

"Built in" = structurally different, not one more layer

All four CVEs we just walked through are controls layered onto one shared kernel. "Built in" means the isolation boundary isn't "trust the host kernel to enforce N separate policies correctly forever." It's structurally different from the start. [ground this in your own current technical positioning]

"You can't bolt on isolation. You have to build it in."



WHAT ACCEPTANCE LOOKS LIKE ORGANIZATIONALLY

> We stopped asking 'is this patched' and started asking 'what's actually the boundary here.'

> The relief wasn't that it got easier. It's that we stopped pretending.

Landing point: acceptance isn't resignation -- it's clarity. Name what K8s isolation is actually for, stop pretending it's something it's not, design around the real boundary.

THE TAKEAWAY

ISOLATION ISN'T A CHECKBOX YOU TICK.

IT'S AN ARCHITECTURE DECISION YOU MAKE ON PURPOSE.

CLOSE



THE TECHNICAL BOTTOM LINE

"Every one of these bugs was in code that had nothing to do with 'container security' when it was written -- that's what sharing a kernel actually costs you."

CLOSE



LANDING IT

“Every hand that went up earlier was right to go up. The only mistake was thinking you were the only one still working through it.”

COME FIND US AFTER THE TALK -- AND TELL US WHICH STAGE YOU'RE IN RIGHT NOW.

QUESTIONS?

FIVE STAGES OF KUBERNETES SECURITY GRIEF

INSERT COIN TO CONTINUE

APPENDIX -- SOURCES

CVES REFERENCED IN THIS TALK

CVE-2022-0492	cgroup v1 release_agent escape via unshare() -- wrong-namespace capability check	oss-security · Unit 42 · Sysdig
CVE-2023-32233	Netfilter nf_tables use-after-free -- unprivileged userns -> kernel r/w	Ubuntu · oss-security · Red Hat
CVE-2019-5736	runc /proc/self/exe breakout -- host runc binary overwrite	Unit 42 · CVE.org
CVE-2024-21626	"Leaky Vessels" -- runc process.cwd() / leaked fd container breakout	Snyk Labs · Wiz

Pulled from vendor write-ups and oss-security threads, not the kernel patch/commit text itself -- worth a technical eyeball before quoting mechanisms live.