

FROM CATASTROPHE TO CHAOS IN PRODUCTION

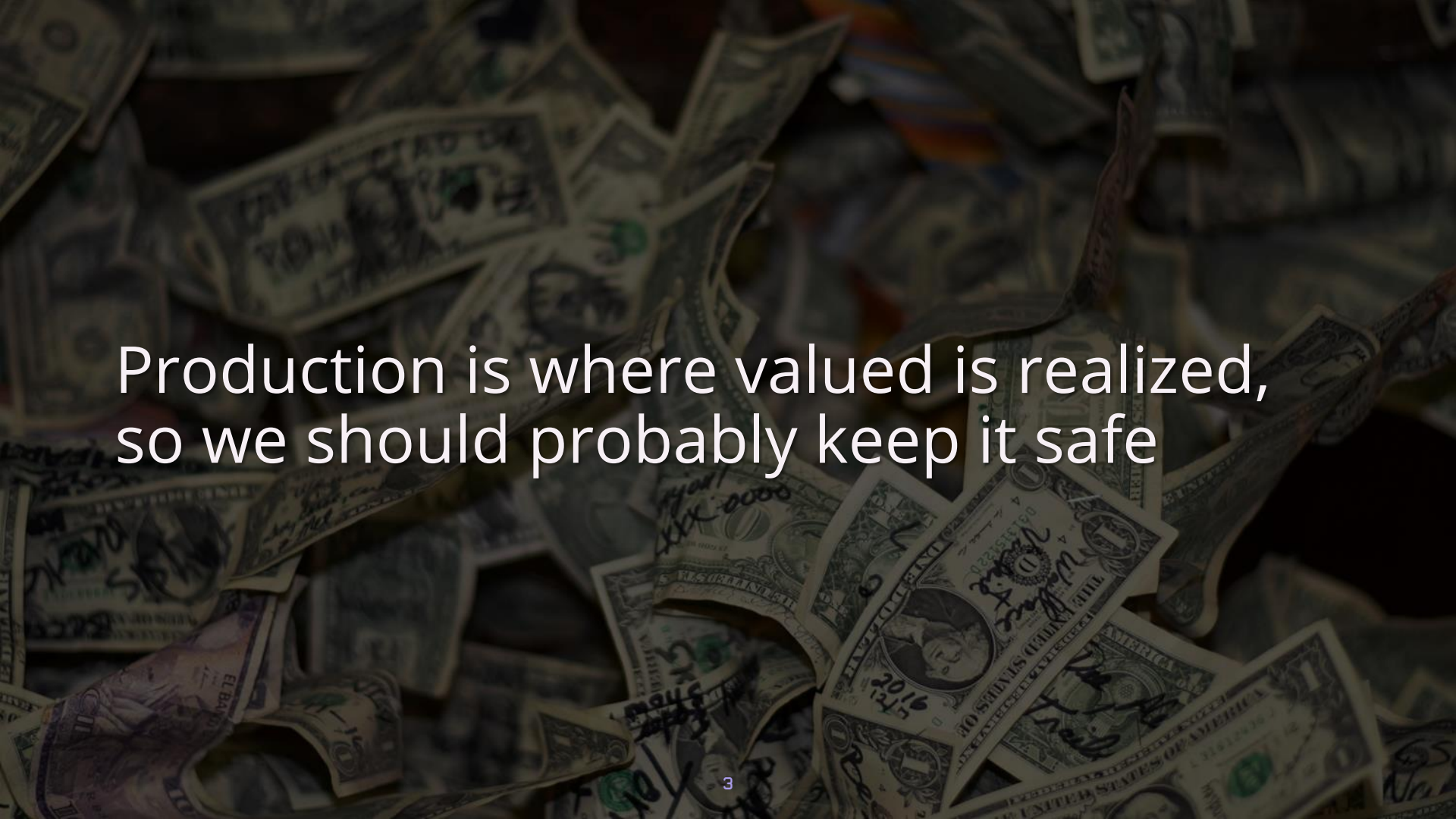
Kelly Shortridge (@swagitda_)

GOTopia Chaos Engineering Day 2021

A black cat's face is centered in the frame, with its eyes glowing a bright, ethereal blue. The cat's fur is dark and textured, and its gaze is directed straight at the viewer.

Hi, I'm Kelly

CAPSULE8



Production is where value is realized,
so we should probably keep it safe



Failure in production feels frightening

A dark, blue-toned image of a staircase. A glowing blue line, resembling a path or a signal, starts from the bottom left and moves upwards, curving and zig-zagging across the steps. The text "...but it doesn't have to end in disaster" is overlaid in white, sans-serif font in the center of the image.

...but it doesn't have to end in disaster



How can we harness failure as a learning opportunity to make production safer?

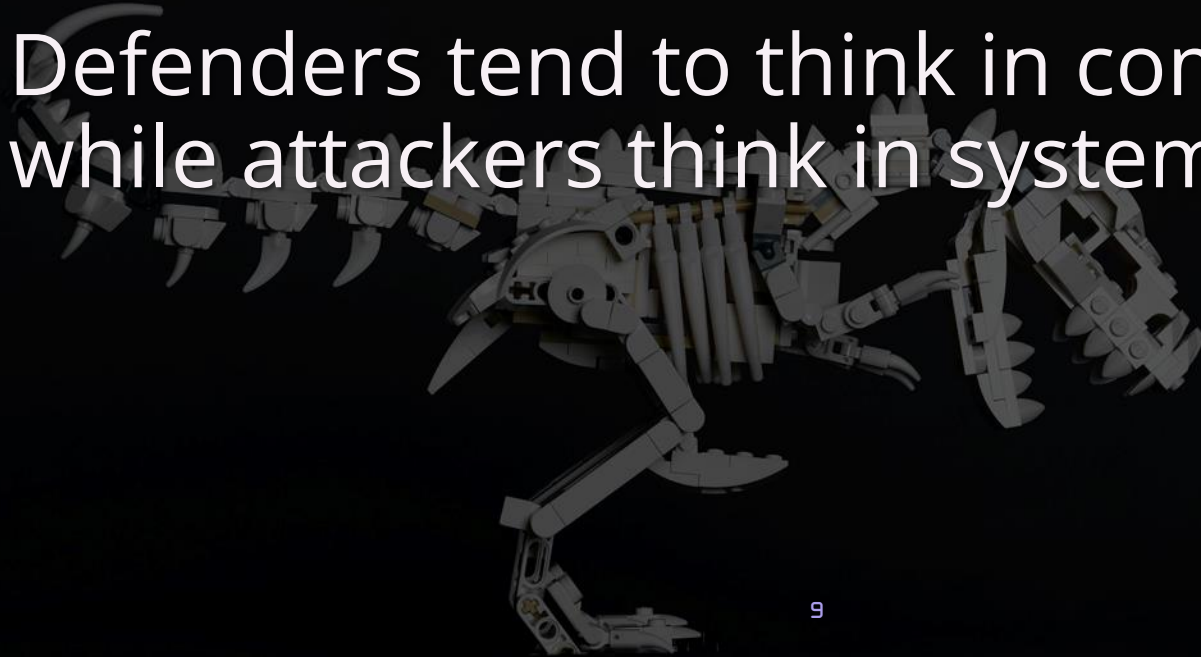
I. Failure in Production

II. Security Chaos
Engineering in Production

A close-up of a lit sparkler against a dark background, with numerous bright sparks radiating outwards. The sparks are primarily orange and yellow, with some appearing as long, thin streaks and others as small, star-like bursts.

I. Failure in Production

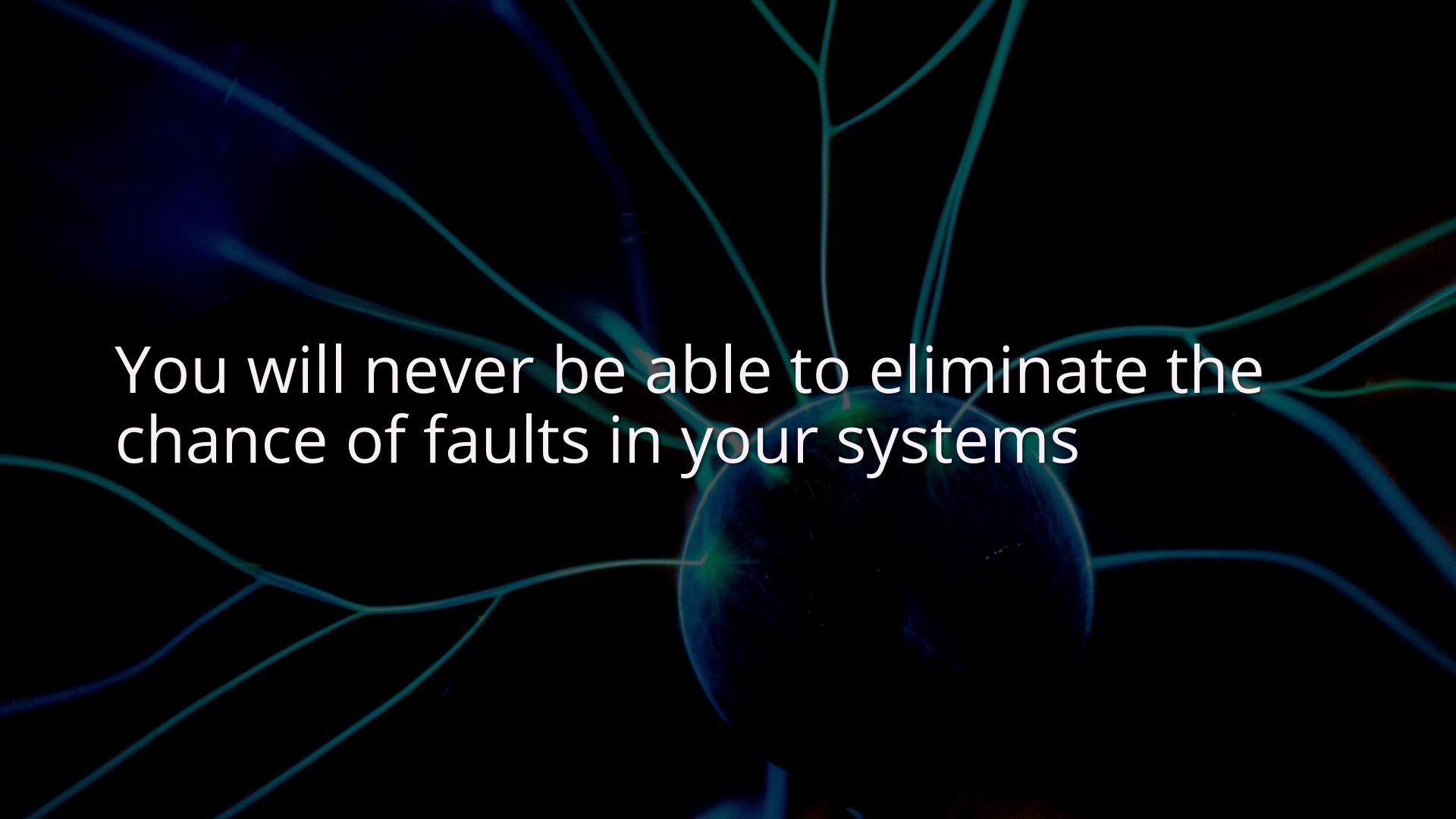
Defenders tend to think in components
while attackers think in systems



Component-level vs. system-level –
faults are different than failures

Faults: “one component of the system deviating from its spec”

Failure: “the system as a whole stops providing the required service to users”

The background of the slide features a dark, almost black, space-like environment. In the center, there is a glowing blue sphere representing the Earth, showing some cloud patterns. From this central sphere, numerous thin, bright blue lines radiate outwards in various directions, resembling a network or a starburst pattern. The lines vary in length and thickness, creating a sense of dynamic energy and connectivity.

You will never be able to eliminate the
chance of faults in your systems

Prevention only goes so far; too many variables are out of your control

A perfectly patched container can still be pwned if there's anon access in K8s

Scan all the code for vulns... then
attackers compromise the code scanner

Yubikeys for GitHub... then attackers
abuse Jenkin's anon script console



Failure in production manifests in a
mess of multiplicitous manners

Private / public clouds, VPS, VMs,
containers, serverless, computerless...



Production environments are complex
systems full of interrelated components

Failure is like a tapestry of interwoven strands that can spread fire to the rest

The background is a dark, abstract composition. In the center, a translucent globe of the Earth is visible, showing continents and oceans. Overlaid on and around the globe are numerous bright, colorful light trails in shades of blue, green, and red. These trails are mostly horizontal and diagonal, with some forming loops or spirals, creating a sense of rapid movement and complexity. The overall effect is one of a busy, high-tech environment.

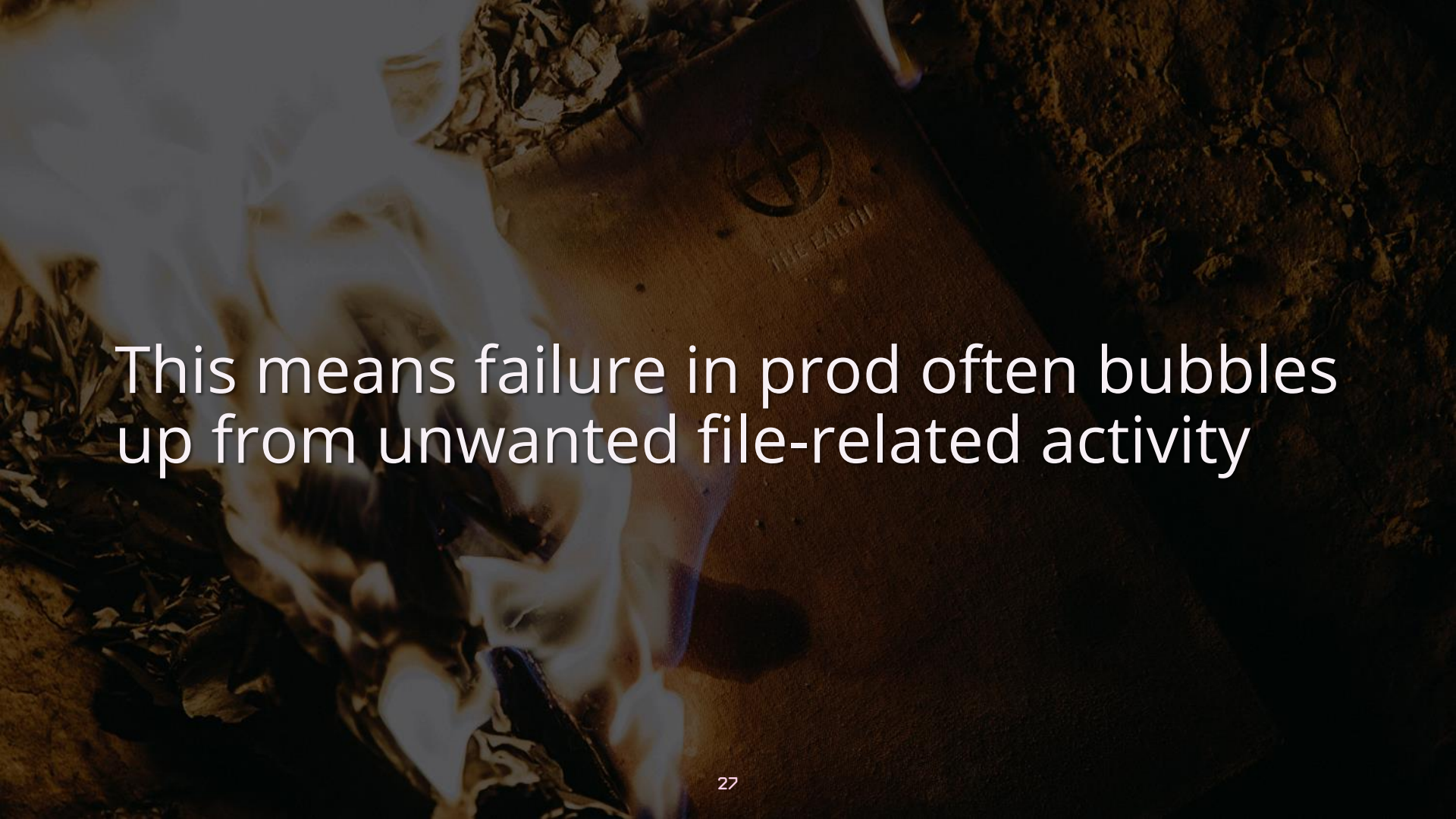
There is a dizzying array of activity that
can jeopardize production operations

Two key types: deliberately malicious (attackers) + accidentally careless (devs)

Sometimes they overlap! Like attaching a debugger to a prod system

Attackers with privileged creds &
“insider threats” are basically the same

Most prod infrastructure runs on Linux,
where everything is a file

The background is a dark, textured surface, possibly a piece of wood or stone. In the upper left, there is a brown paper bag with a circular logo containing a cross-like symbol and the text 'THE EARTH' below it. The bag is partially covered by crumpled white paper and some dried, brown leaves or twigs. The overall lighting is dim and moody.

This means failure in prod often bubbles
up from unwanted file-related activity

Example 1: Log files are deleted or tampered – your ops is likely screwed

Example 2: Changes to boot files, root cert stores, or SSH keys – stability snafus


Example 3: Resource limits are disabled
– highly sus and doubtless disastrous

Confronted with such complexity, how can we constructively cope?



II. Security Chaos Engineering in Prod

Our goal is to prevent faults from causing failures as much as we can

A black and white photograph showing a sequence of water droplets falling into a pool of water. The droplets are captured in mid-air, creating a sense of motion. Below the droplets, concentric ripples spread out across the surface of the water. The background is dark and out of focus.

Purposefully triggering faults lets you
realize and test your success towards it

Security Chaos Engineering: Let's harness failure to build knowledge

Conducting experiments generates evidence & builds muscle memory

Make incident response boring because
it feels routine after repeated practice



SCE untangles relations between prod
components to curtail contagion

Learning how your systems respond to failure requires testing in prod itself

...but you can start in staging to build confidence before migrating to prod

The background of the slide is a dark blue field filled with intricate, colorful patterns. These patterns consist of numerous overlapping, swirling lines in shades of green, purple, and blue. Interspersed among these lines are several spherical shapes, each composed of concentric, intersecting lines that create a complex, web-like structure. The overall effect is a dynamic and visually rich abstract composition.

What SCE experiments should you try?

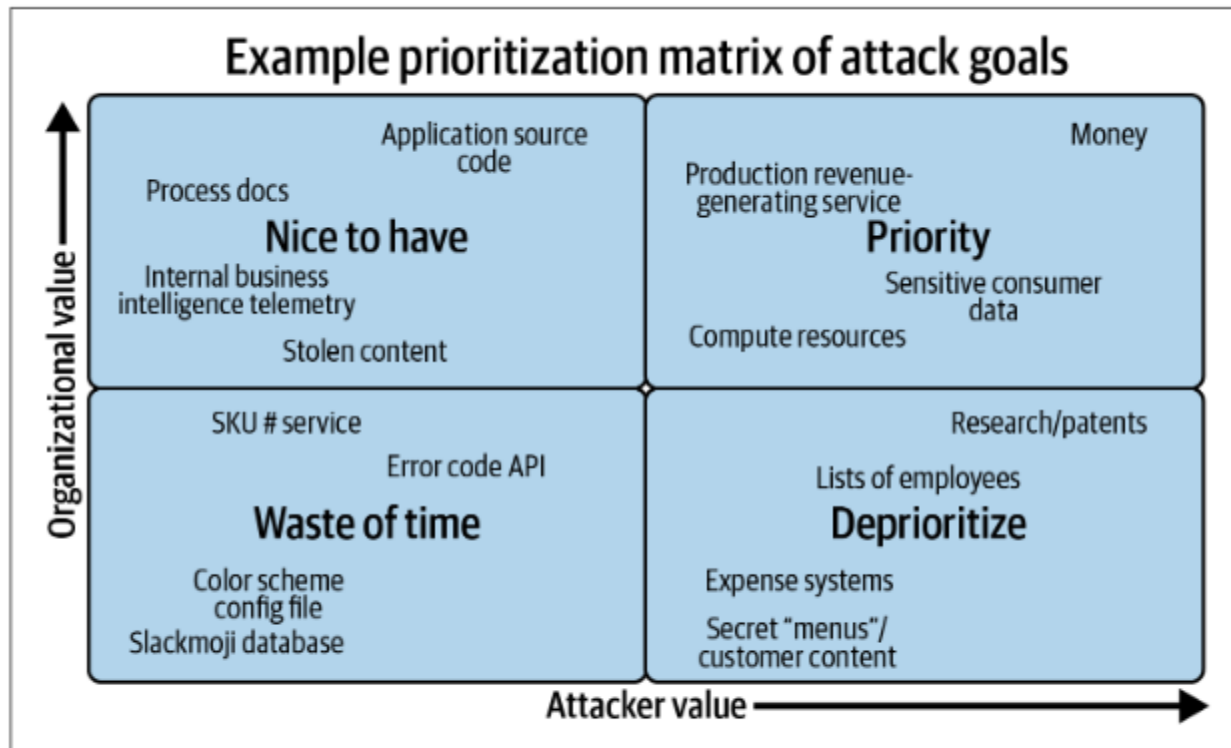


Figure 2-5. Example prioritization matrix of assets relative to attacker value and organizational value.

Let's explore some examples...



Example 1: Create & execute a new file in a container

How does your container respond to new file exec? Does it affect the cluster?

Example 2: Inject program crashes

Does your node restart itself? How quickly can you redeploy post-crash?



Example 3: Disable resource limits (CPU, file descriptors, memory, restarts, etc.)

Can an infinite script take up resources?
Do slower response times propagate?

The background of the slide is a dark blue field filled with a complex network of glowing lines and dots. The lines are thin and translucent, with a blueish-purple hue, radiating from various points. Interspersed among these lines are numerous small, bright orange-yellow dots, some of which are surrounded by a soft, glowing blue halo. The overall effect is reminiscent of a digital network or a microscopic view of a complex structure.

Example 4: Disable access to DNS

How reliant are your systems on external DNS? Do you have a fallback?

Example 5: Time travel on a host

How do systems handle expired certs?
Do time-related issues bork services?



In Conclusion

The background of the slide is a dark, textured composition. It features wispy, ethereal blue smoke or mist that rises from a bright, glowing point at the bottom center. Scattered throughout the scene are numerous small, bright orange and yellow sparks or particles, some of which appear to be falling or drifting. The overall effect is one of dynamic energy and transformation, set against a deep black backdrop.

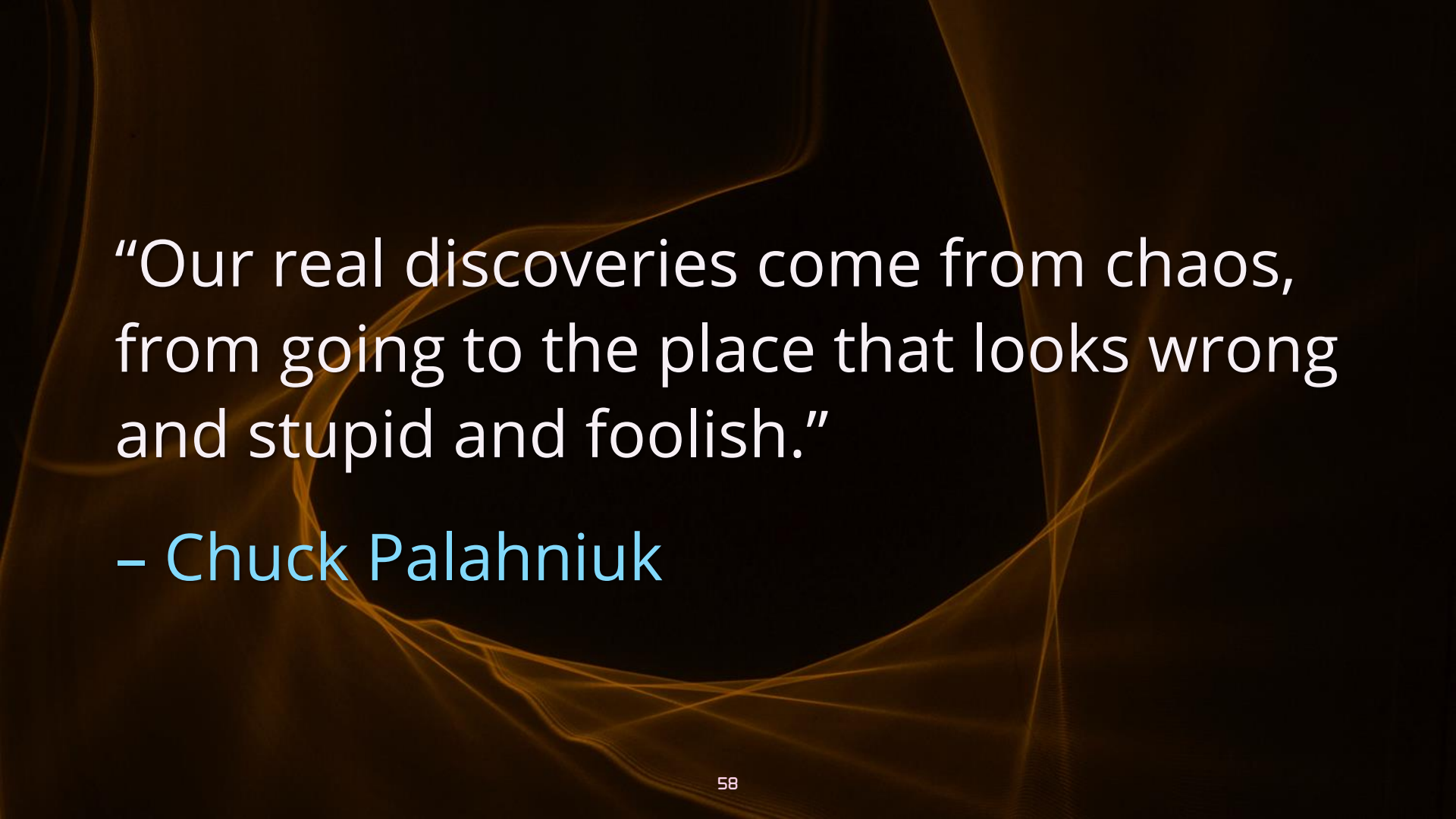
• Failure in production is inevitable, so you must learn from it early and often



Conducting experiments uncovers new
knowledge & builds muscle memory



Security chaos engineering builds
confidence in the safety of prod systems



“Our real discoveries come from chaos,
from going to the place that looks wrong
and stupid and foolish.”

– Chuck Palahniuk

Download for free:
<https://www.verica.io/sce-book/>

O'REILLY®

Security Chaos Engineering

Gaining Confidence in Resilience
and Safety at Speed and Scale

Aaron Rinehart & Kelly Shortridge

REPORT



@swagitda_



/in/kellyshortridge



kelly@shortridge.io