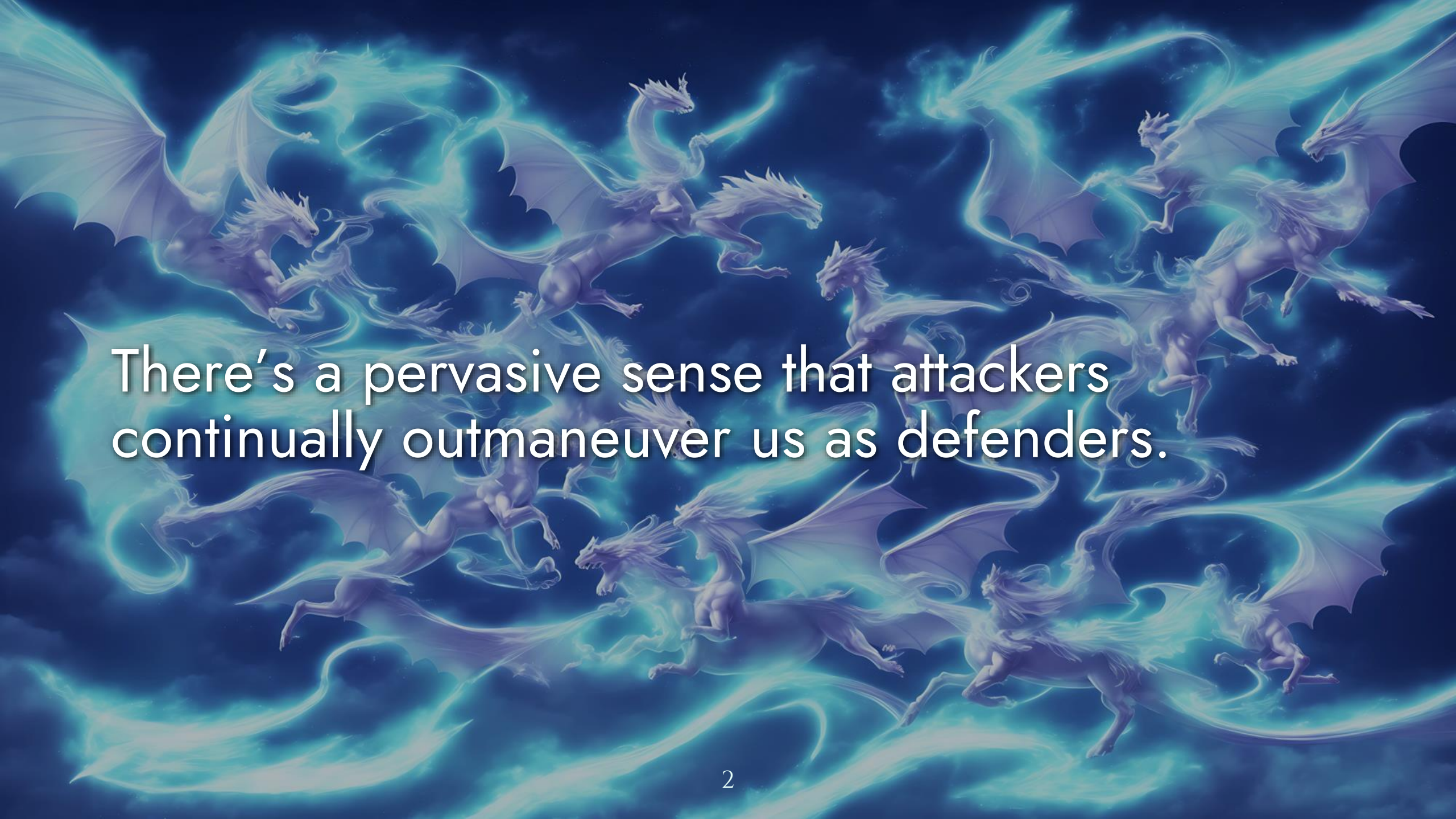
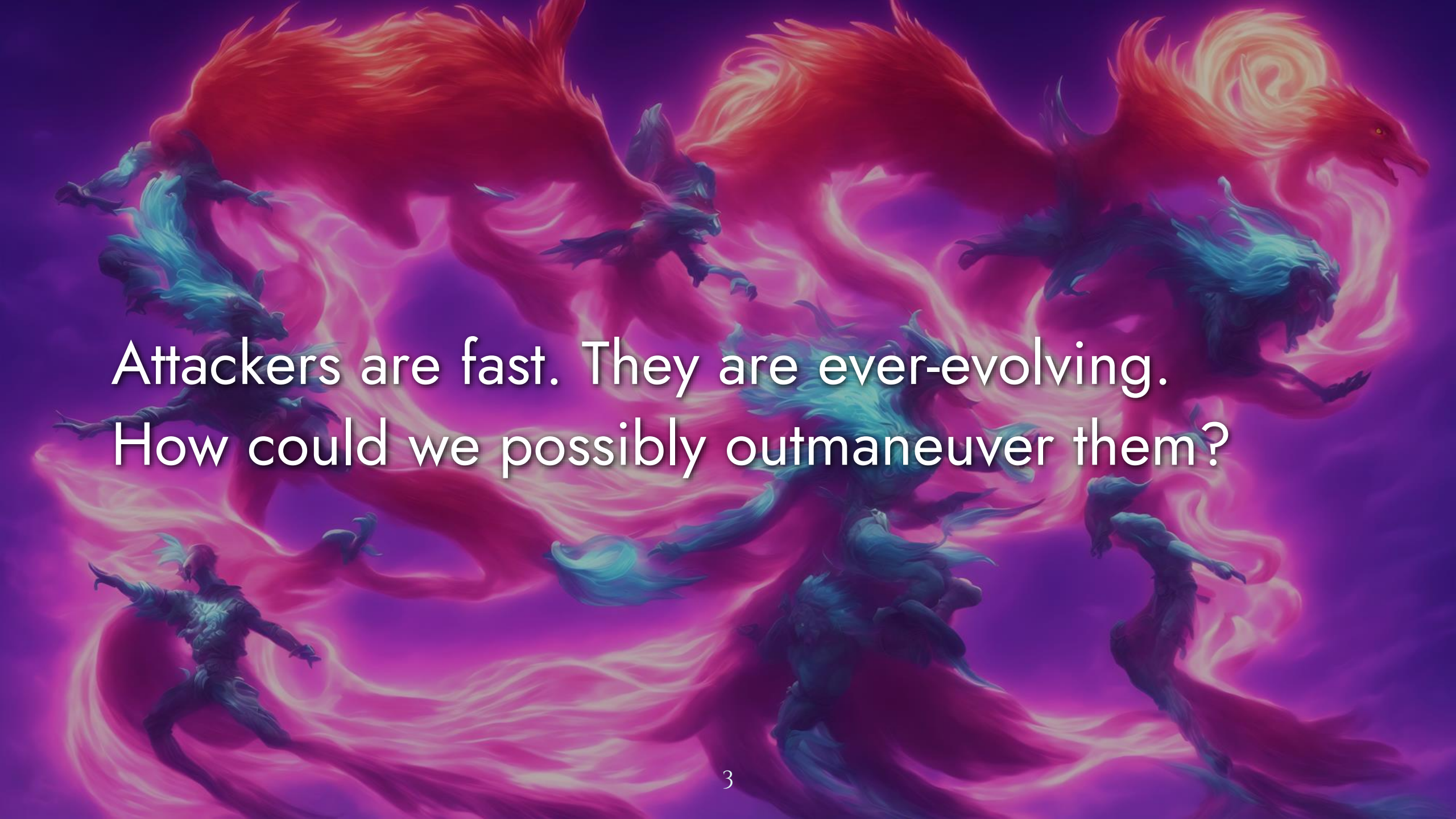
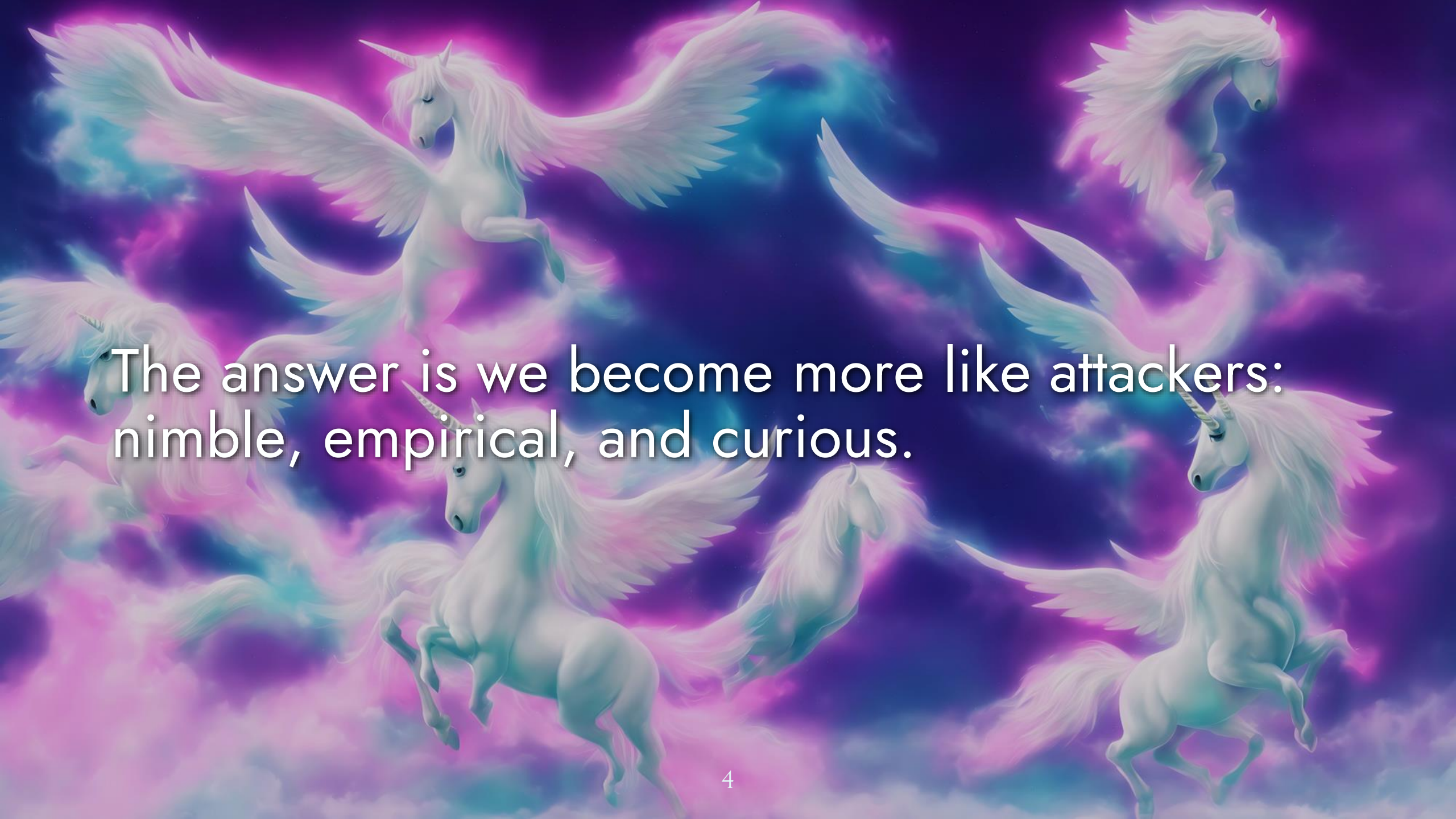
Fast, Ever-evolving Defenders: The Resilience Revolution

Kelly Shortridge @swagitda_ | @shortridge

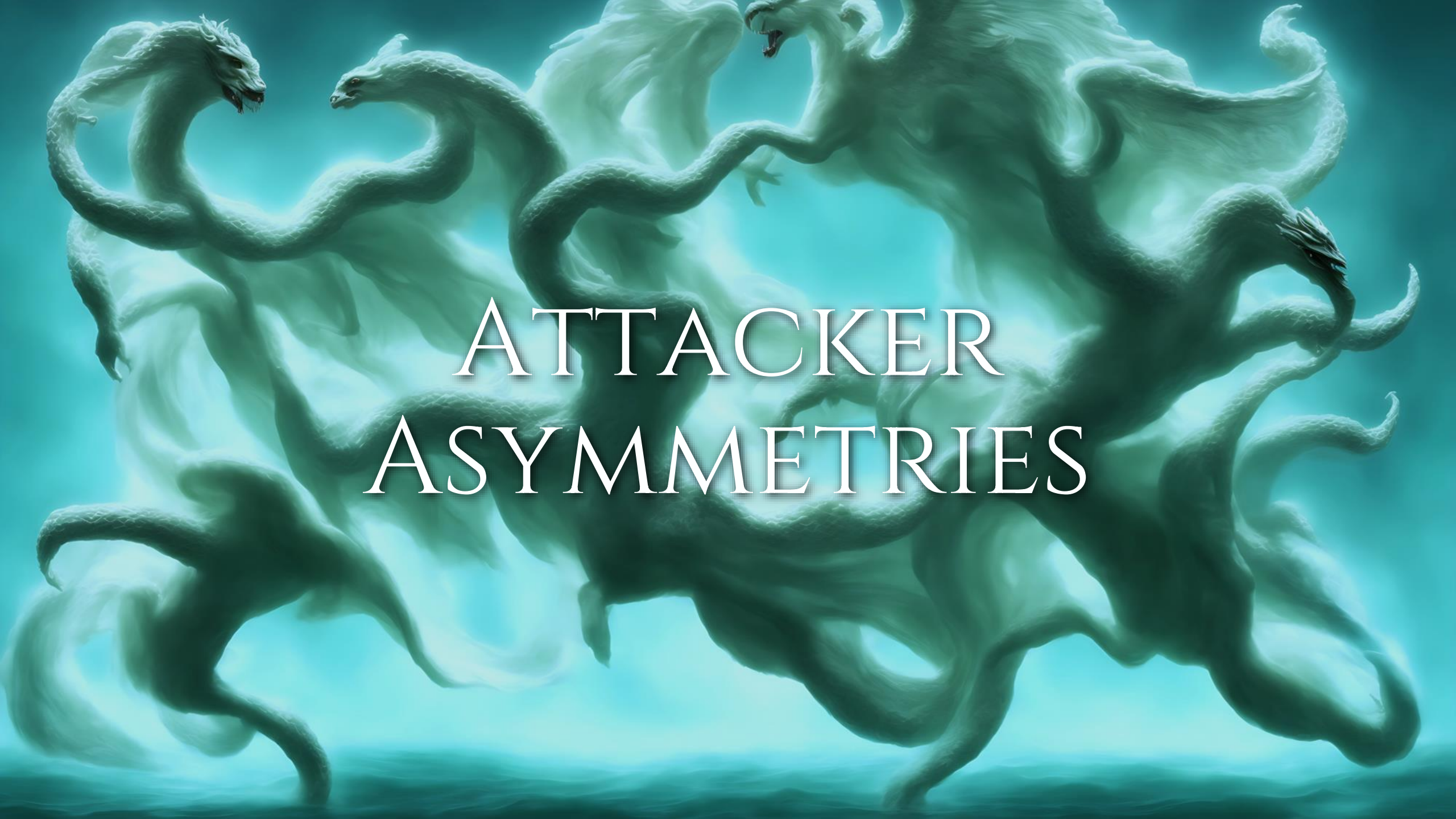
Black Hat USA 2023

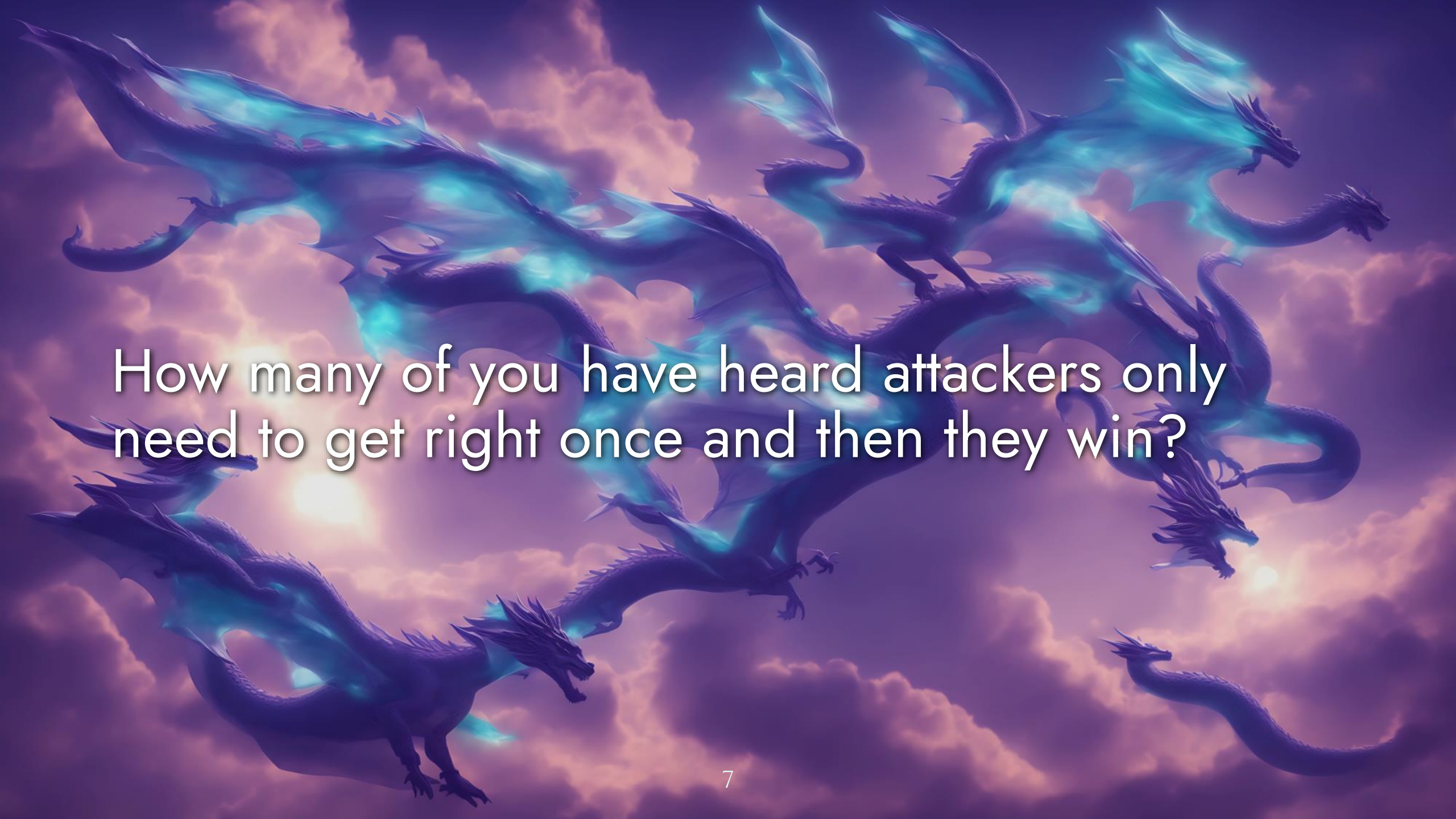

There’s a pervasive sense that attackers
continually outmaneuver us as defenders.


Attackers are fast. They are ever-evolving.
How could we possibly outmaneuver them?


The answer is we become more like attackers:
nimble, empirical, and curious.


This talk is about revolution – a new paradigm for systems defense, grounded in resilience.


Attacker
Asymmetries


How many of you have heard attackers only need to get right once and then they win?

That's a myth. They need to get right once for initial access then get it right *every time* after.

SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_


So, what are attackers' *real* advantages?

1) Attackers have a faster operational tempo

SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_

2) Attackers design, develop, and operate mechanisms to outmaneuver us

SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_

3) Attackers research interconnections and interactions in systems

SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_

4) Attackers have more tangible and actionable success metrics

SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_


There is no reason why we can’t steal these advantages for ourselves as defenders.

All of these reflect a foundation of resilience: the ability to prepare for, recover from, and adapt to adverse events.

SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_


We can seize opportunities that grant us these same advantages via the resilience revolution.


I. Faster Tempo

Attackers pivot quickly in the face of adversity.

SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_


Attackers also rapidly evolve their methods.

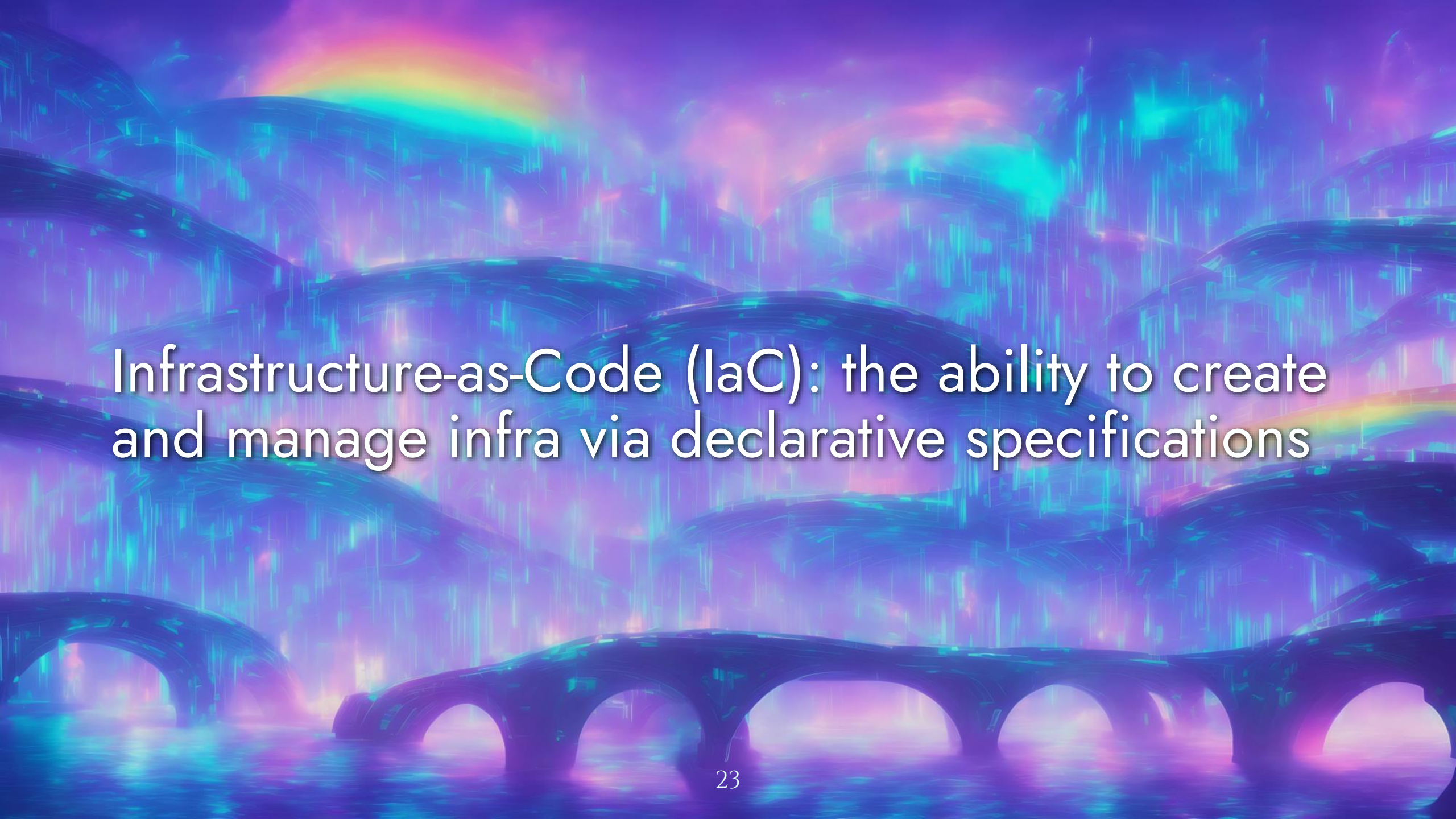
We can achieve a faster tempo by adopting approaches from modern software engineering.

SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_


Configuration as Code

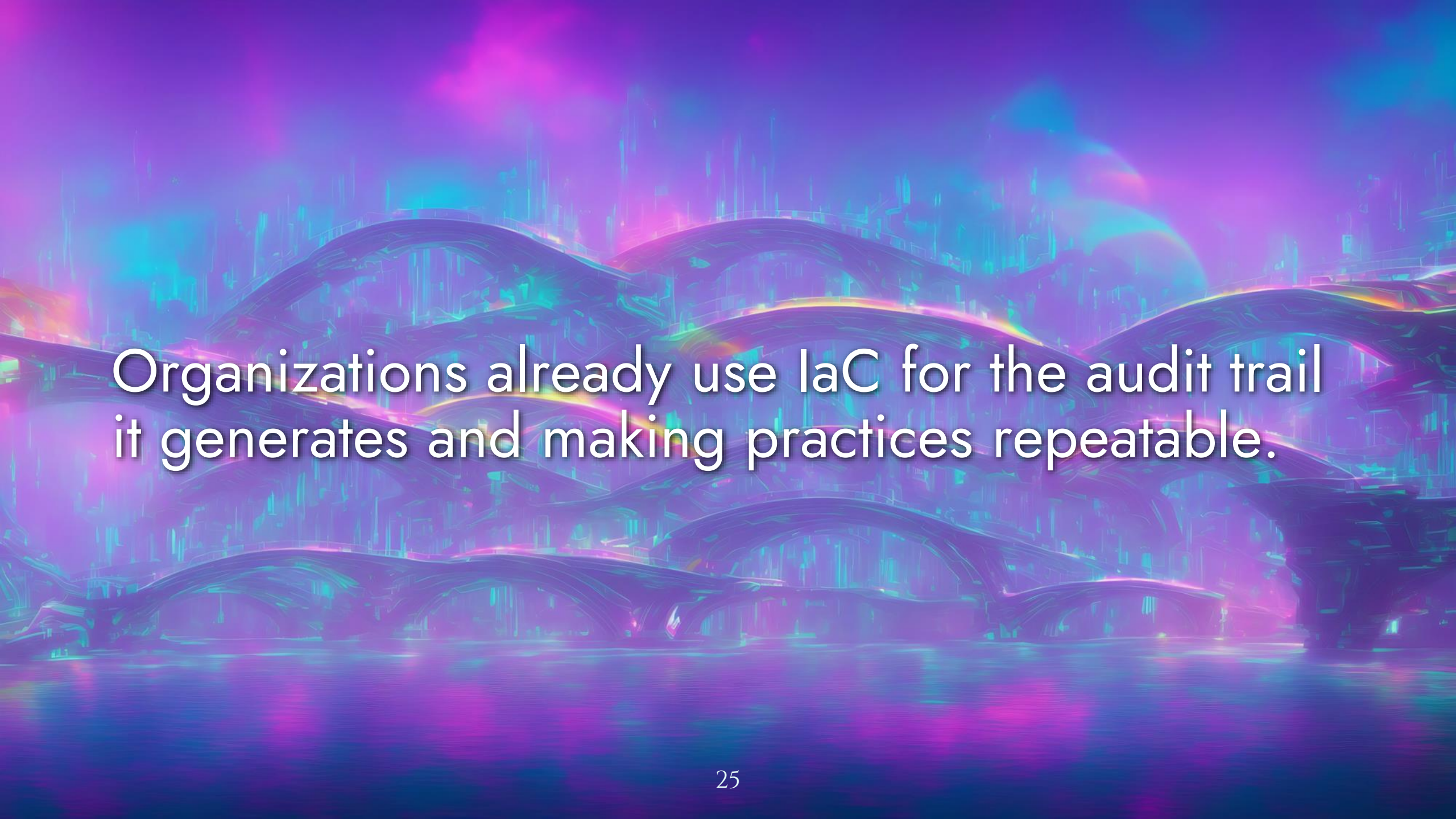
CaC: the practice of declaring configurations through markup rather than manual processes

SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_


Infrastructure-as-Code (IaC): the ability to create and manage infra via declarative specifications

We generate the same environment every time, creating more reliable and predictable services.

SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_


Organizations already use IaC for the audit trail
it generates and making practices repeatable.

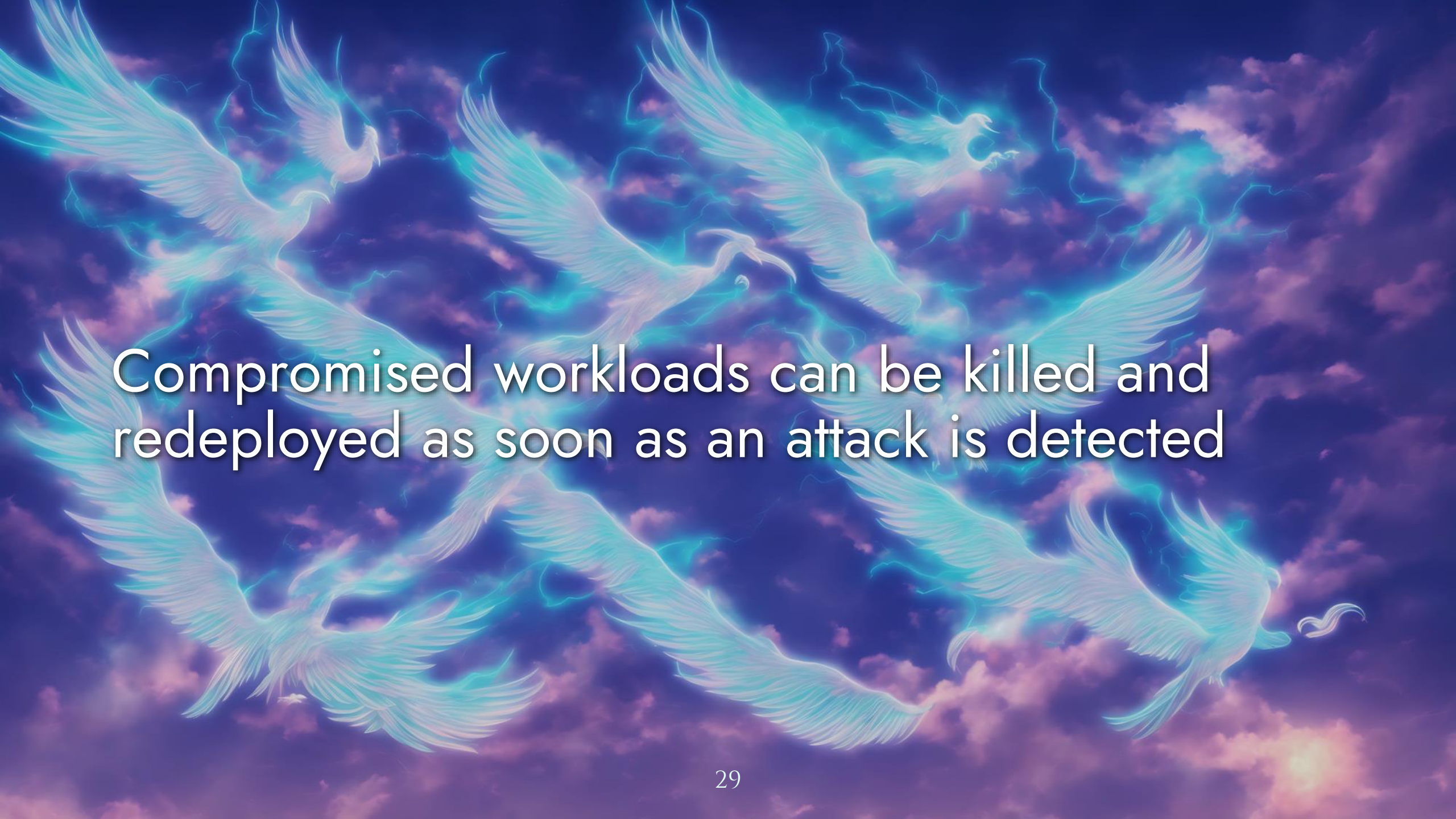
Let's take a whirlwind tour of IaC's bountiful benefits for security programs:

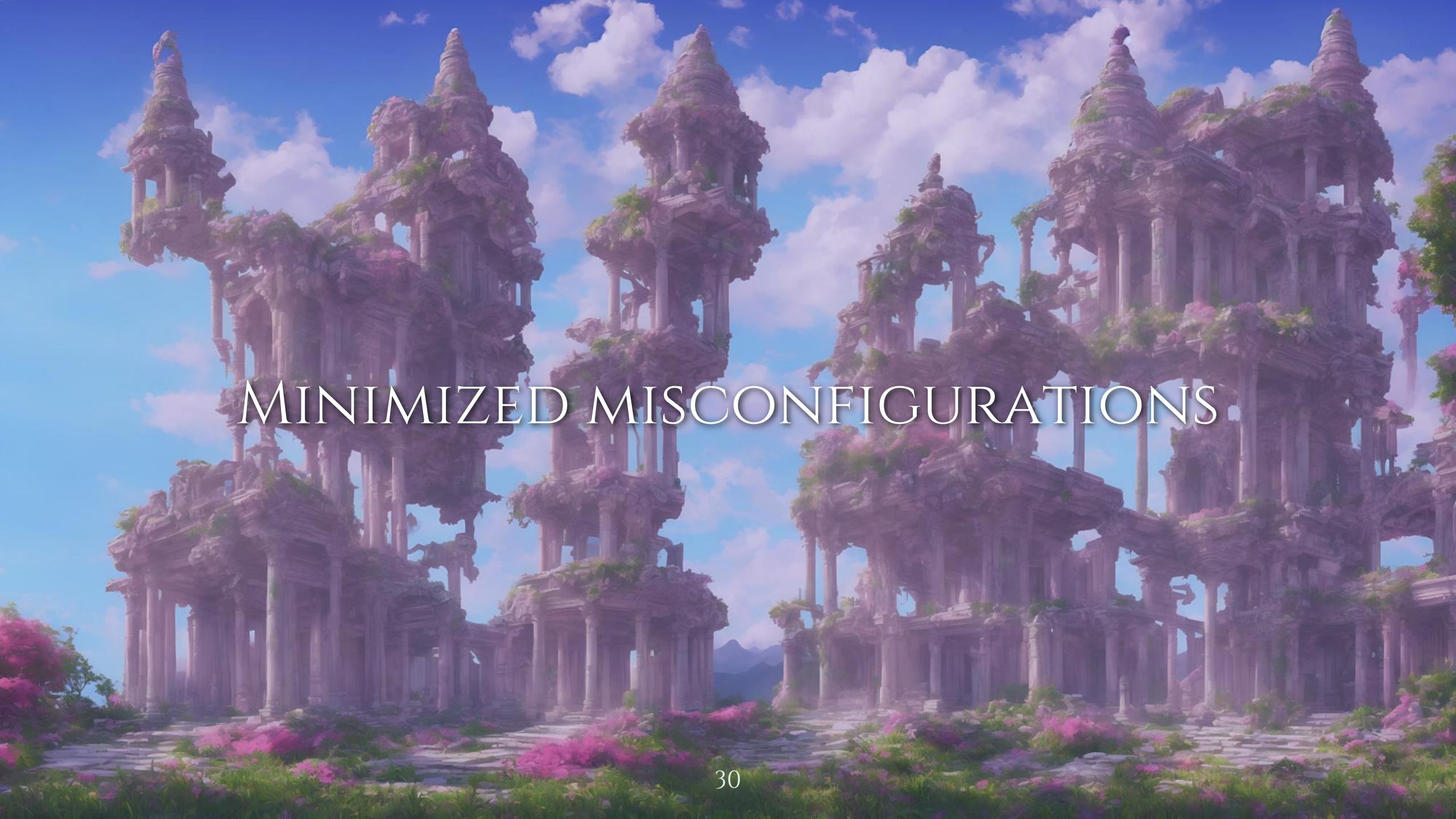
SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_


Faster Incident Response

Automatically redeploy infrastructure when incidents happen… or even leading indicators

SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_


Compromised workloads can be killed and
redeployed as soon as an attack is detected


Minimized misconfigurations

NSA: misconfigurations are the most common cloud vuln; easy to exploit + highly prevalent

SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_


IaC helps correct misconfigurations by users
and automated systems (machines) alike

Faster patching and security changes

The *real* lesson of Equifax: patching processes must be usable, else procrastination is rational

SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_

IaC reduces friction for releasing patches, updates, or fixes & decentralizes the process

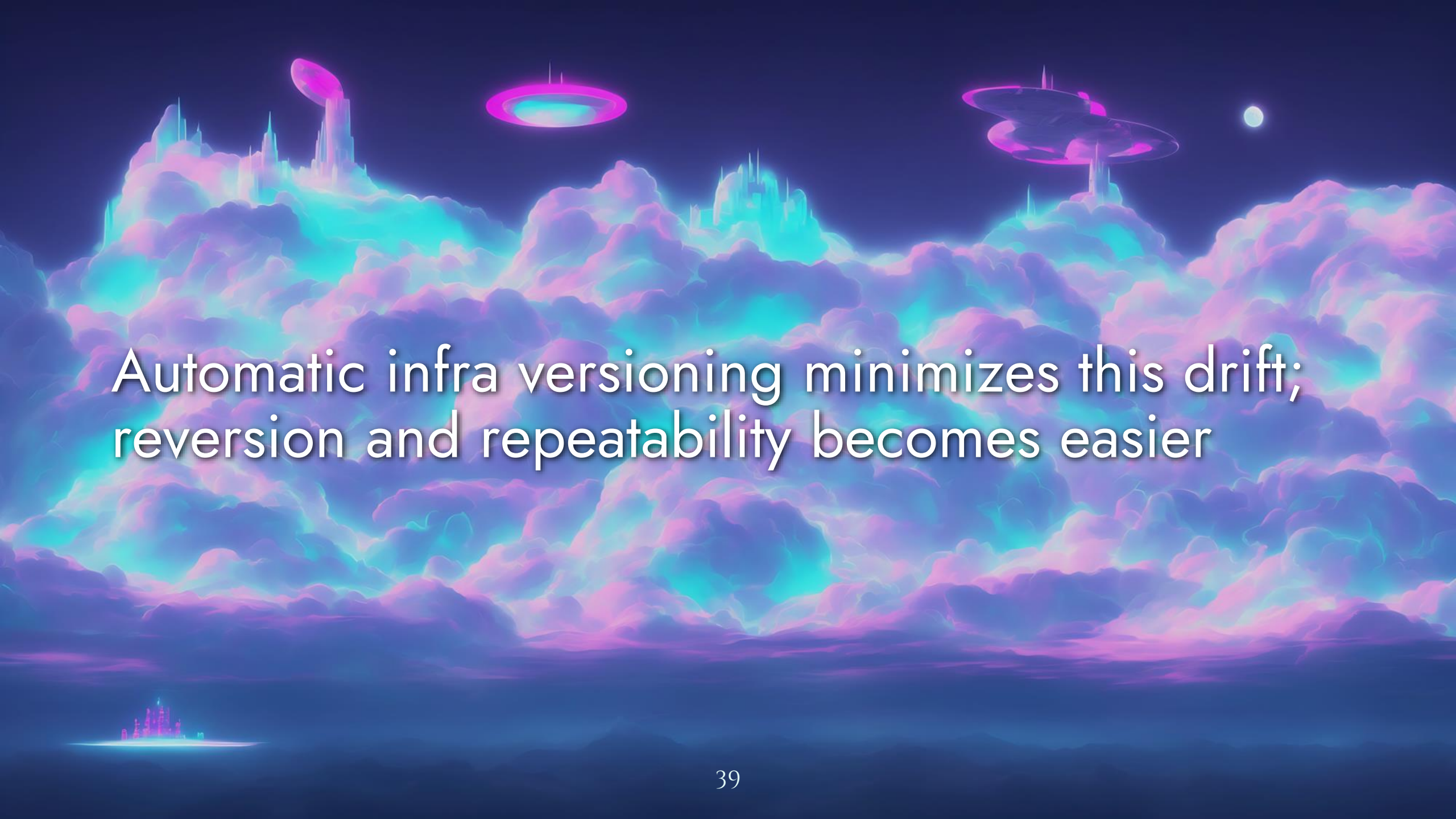
Protip: if an organizational process is unusable or cumbersome, it will be circumvented.

SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_


Minimized
Environmental
Drift

Environmental drift: configs or other attributes "drifting" into an inconsistent state

SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_


Automatic infra versioning minimizes this drift;
reversion and repeatability becomes easier

Catching vulnerable configurations

Status quo is authenticated scanning in production, which introduces new attack paths

SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_


IaC removes that hazard, instead scanning the code files to find vulnerable assets or configs

Stronger change control

IaC introduces change control via SCM, enabling peer reviews on configs + changelog

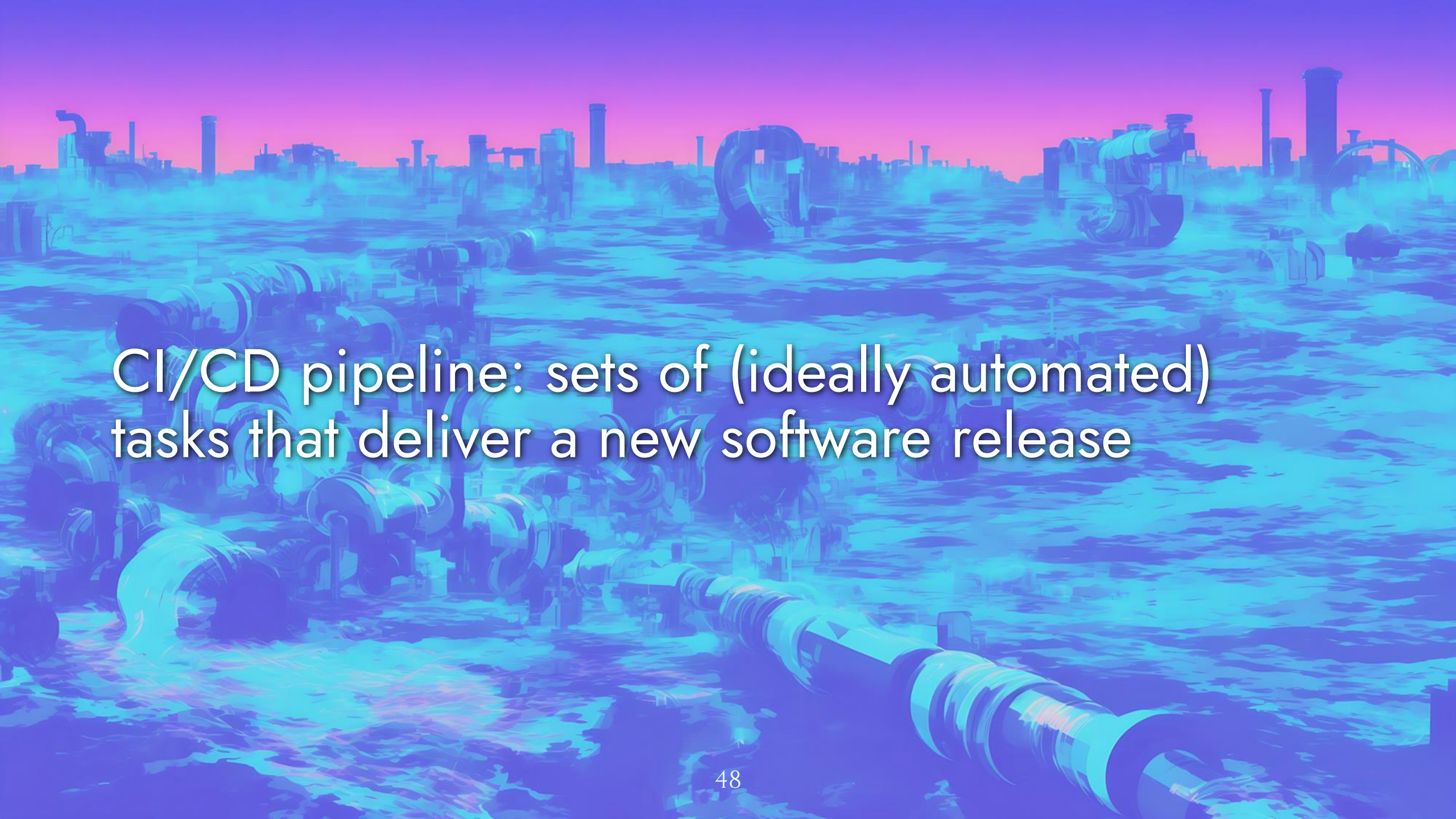
SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_


tl;dr IaC grants us a faster operational tempo in a variety of dimensions


Automating Security
Checks

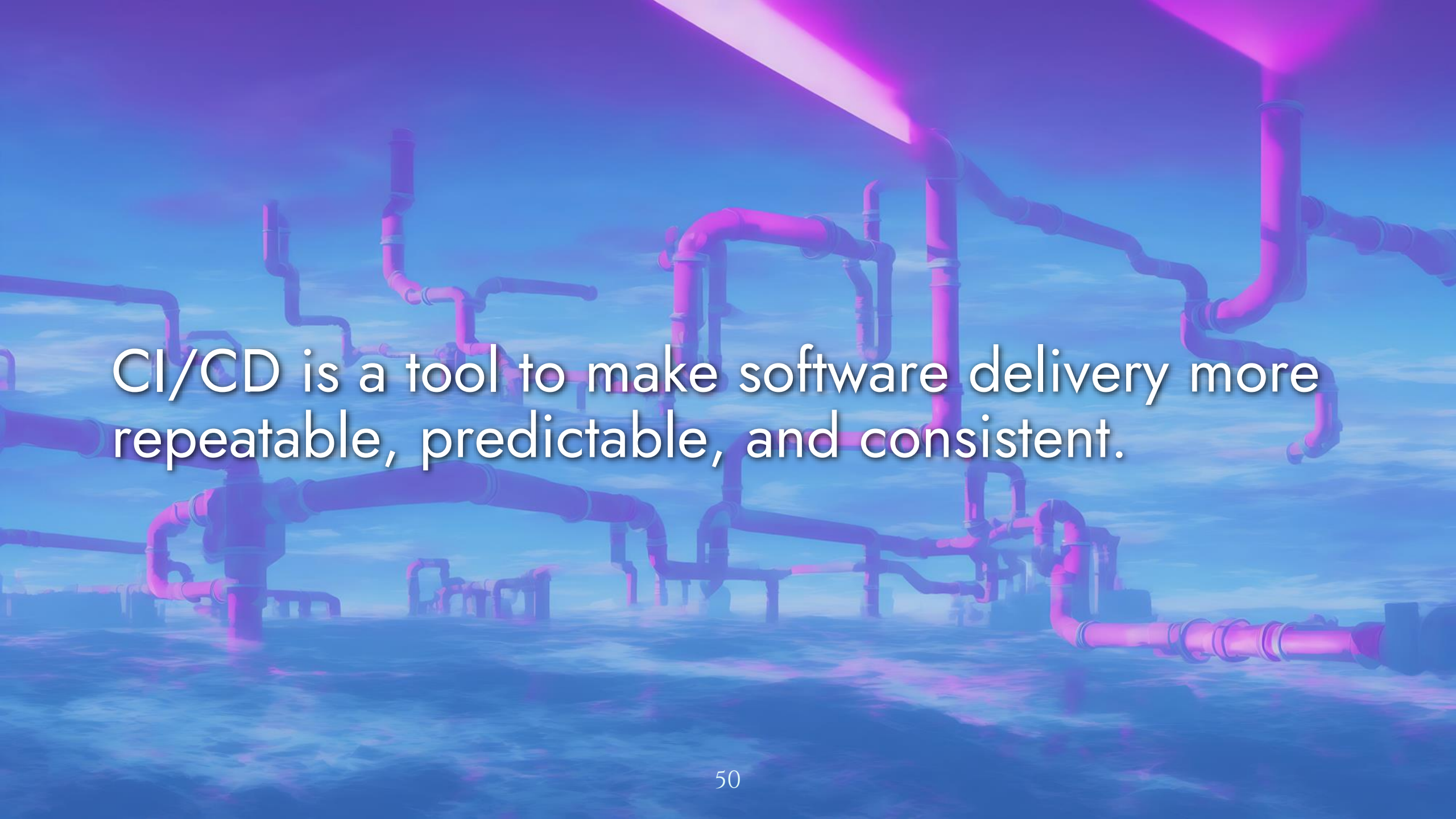
CI/CD accelerates dev and delivery of software features without hurting reliability or quality

SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_


CI/CD pipeline: sets of (ideally automated)
tasks that deliver a new software release

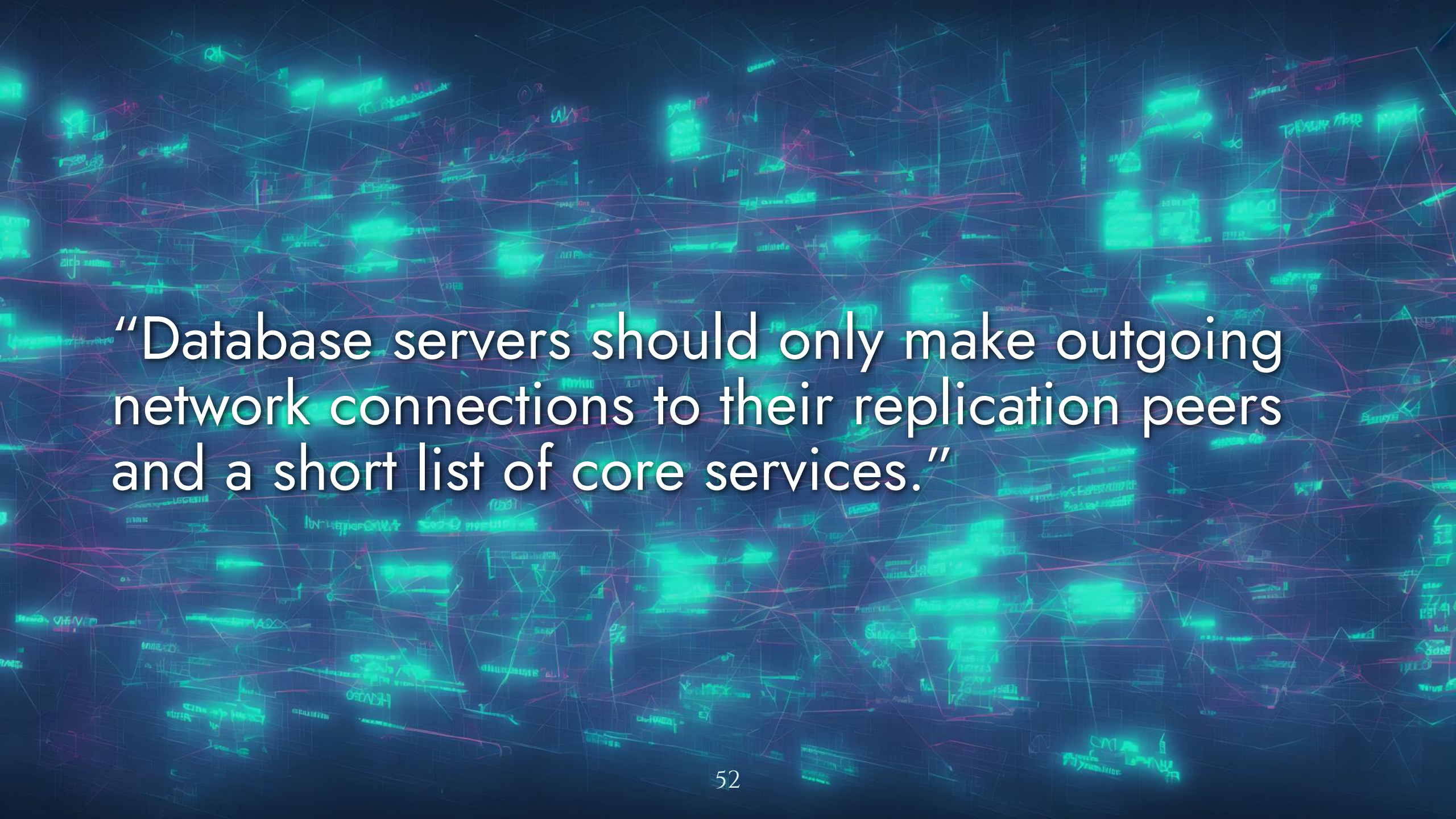
Compiling the app (building) + testing code + deploying to test/staging + delivering to prod

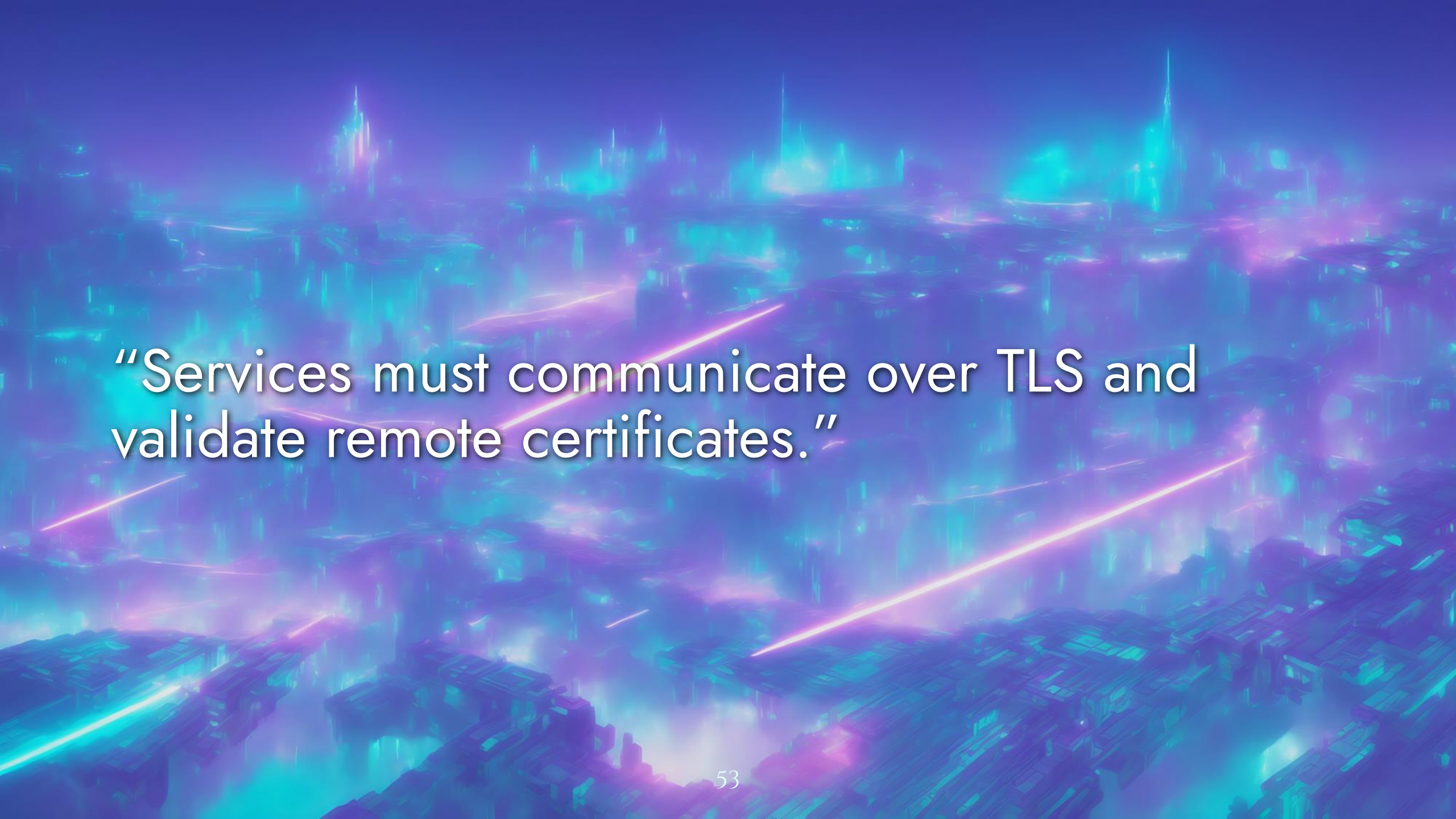
SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_


CI/CD is a tool to make software delivery more repeatable, predictable, and consistent.

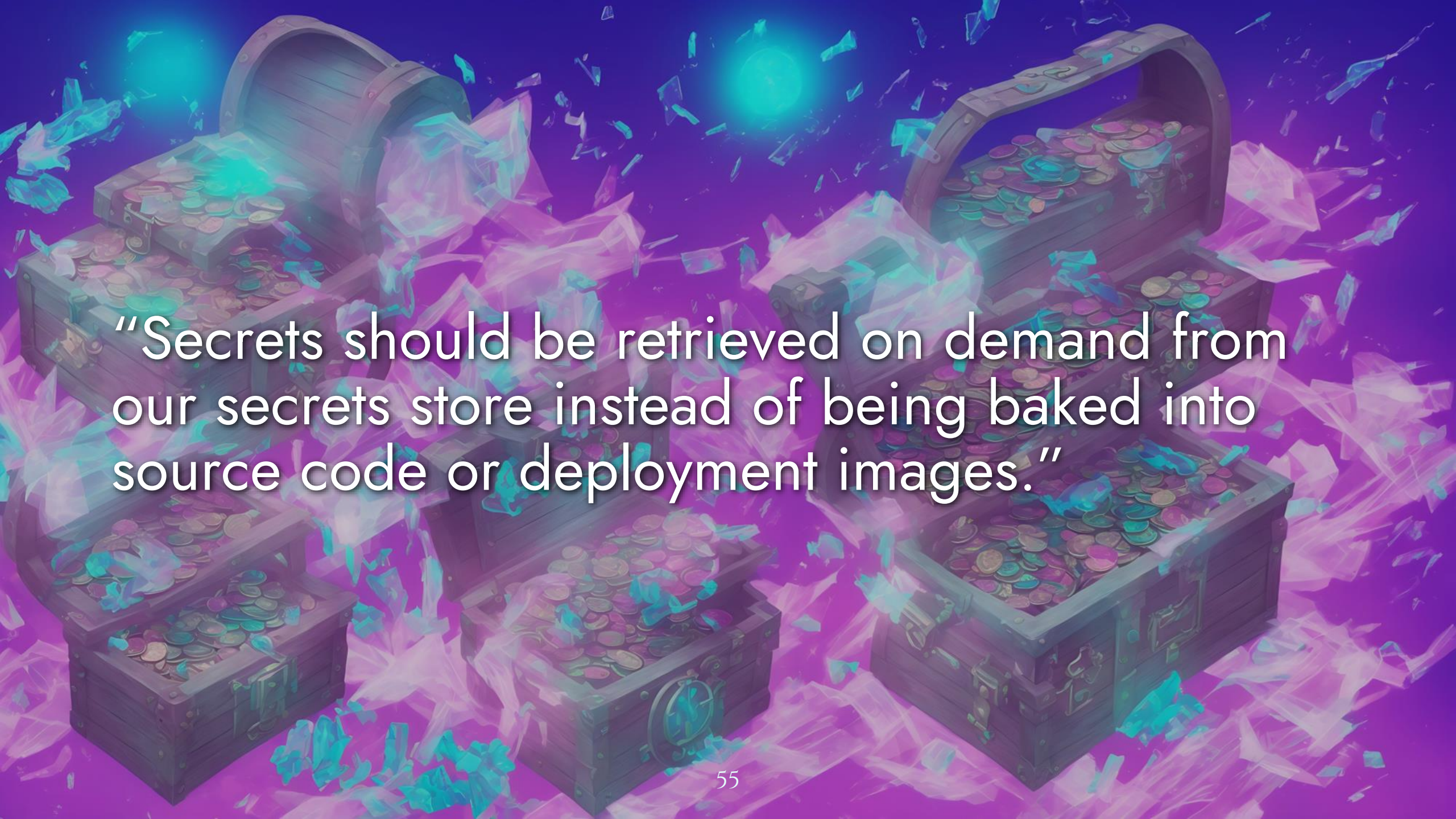
We can enforce invariants: achieve properties we want every time we build + deploy + deliver

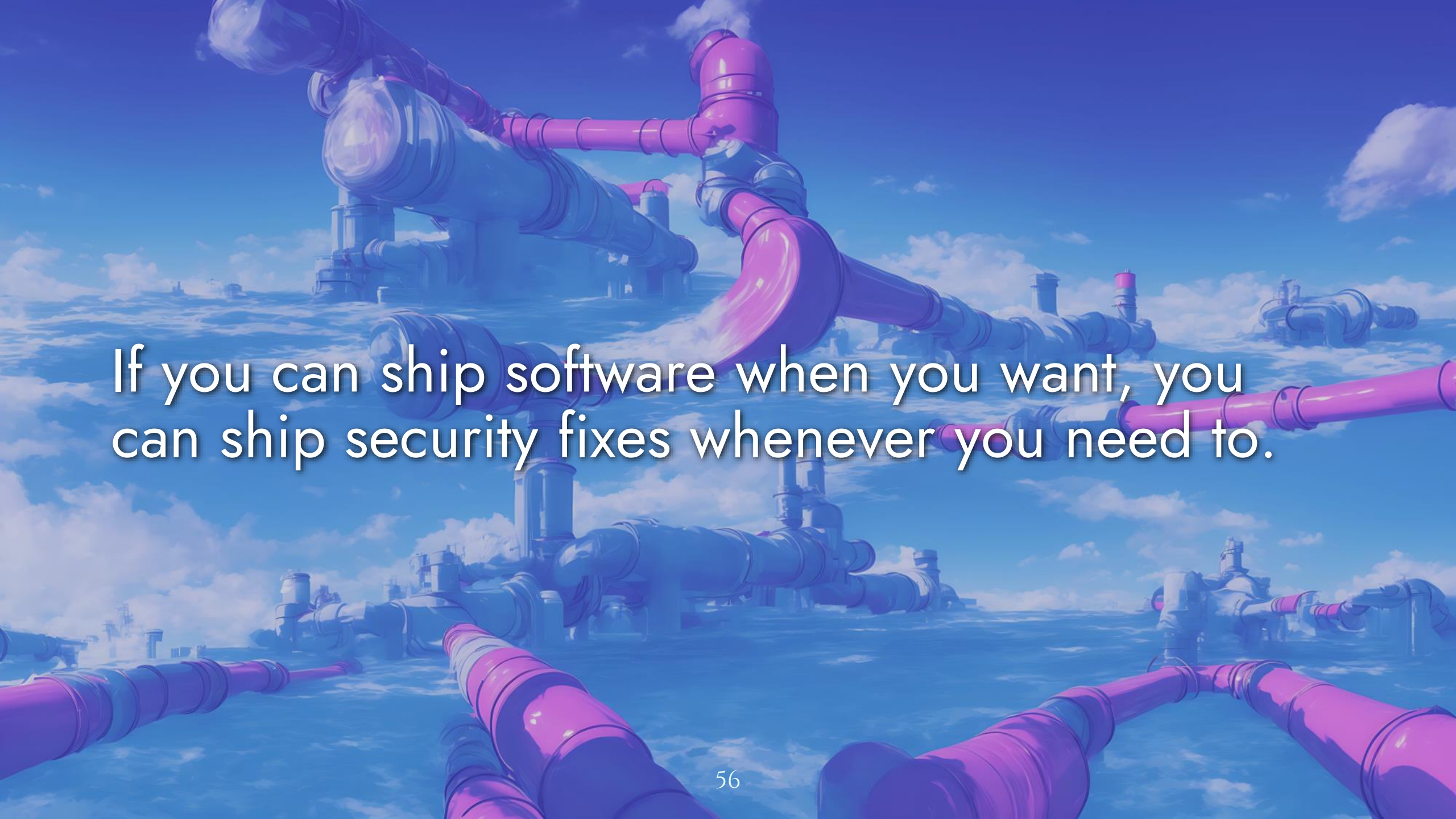
SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_


“Database servers should only make outgoing network connections to their replication peers and a short list of core services.”


“Services must communicate over TLS and validate remote certificates.”

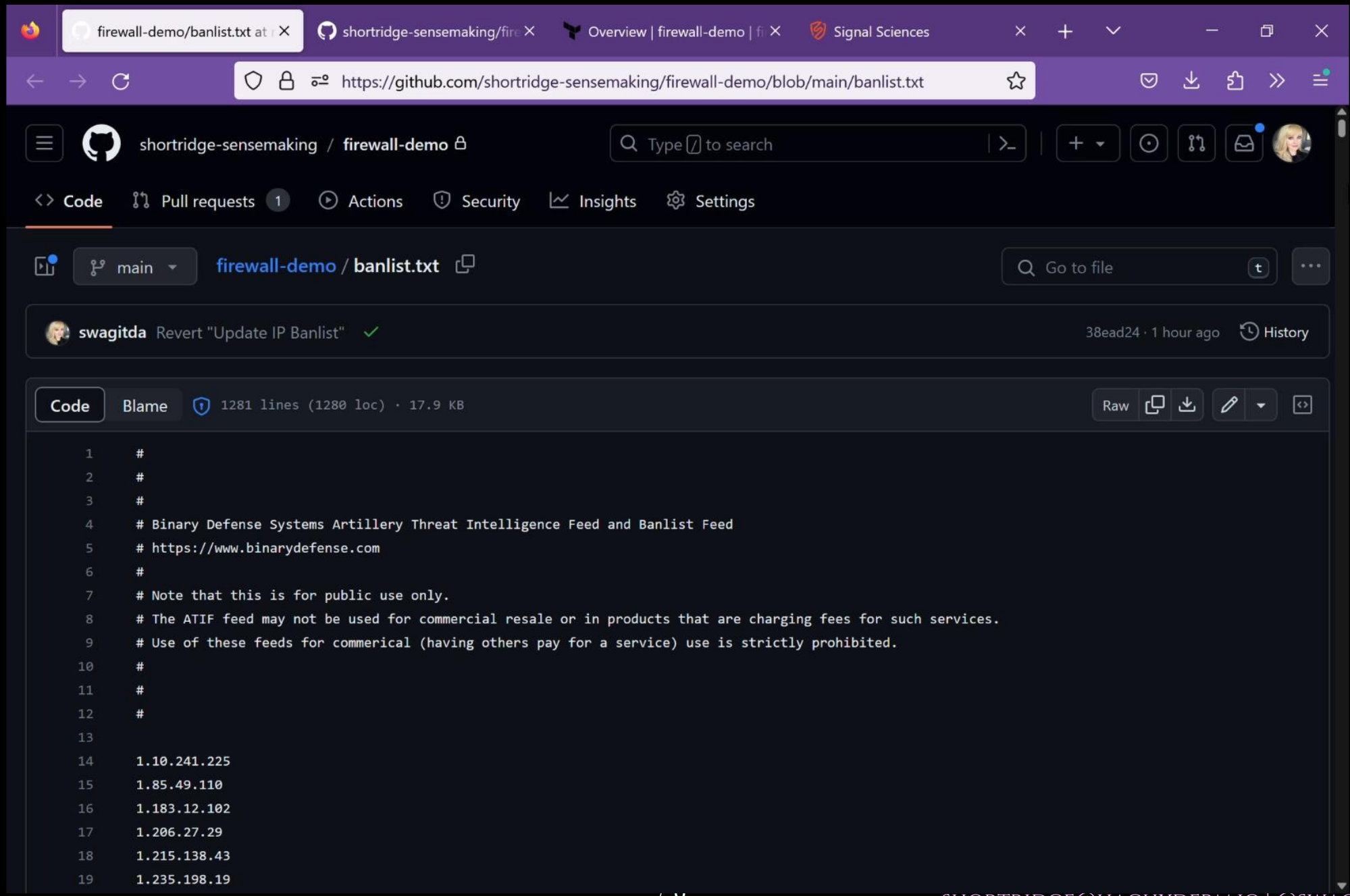

“Only images built by our CI/CD system may run on the production Kubernetes cluster.”


“Secrets should be retrieved on demand from our secrets store instead of being baked into source code or deployment images.”


If you can ship software when you want, you can ship security fixes whenever you need to.

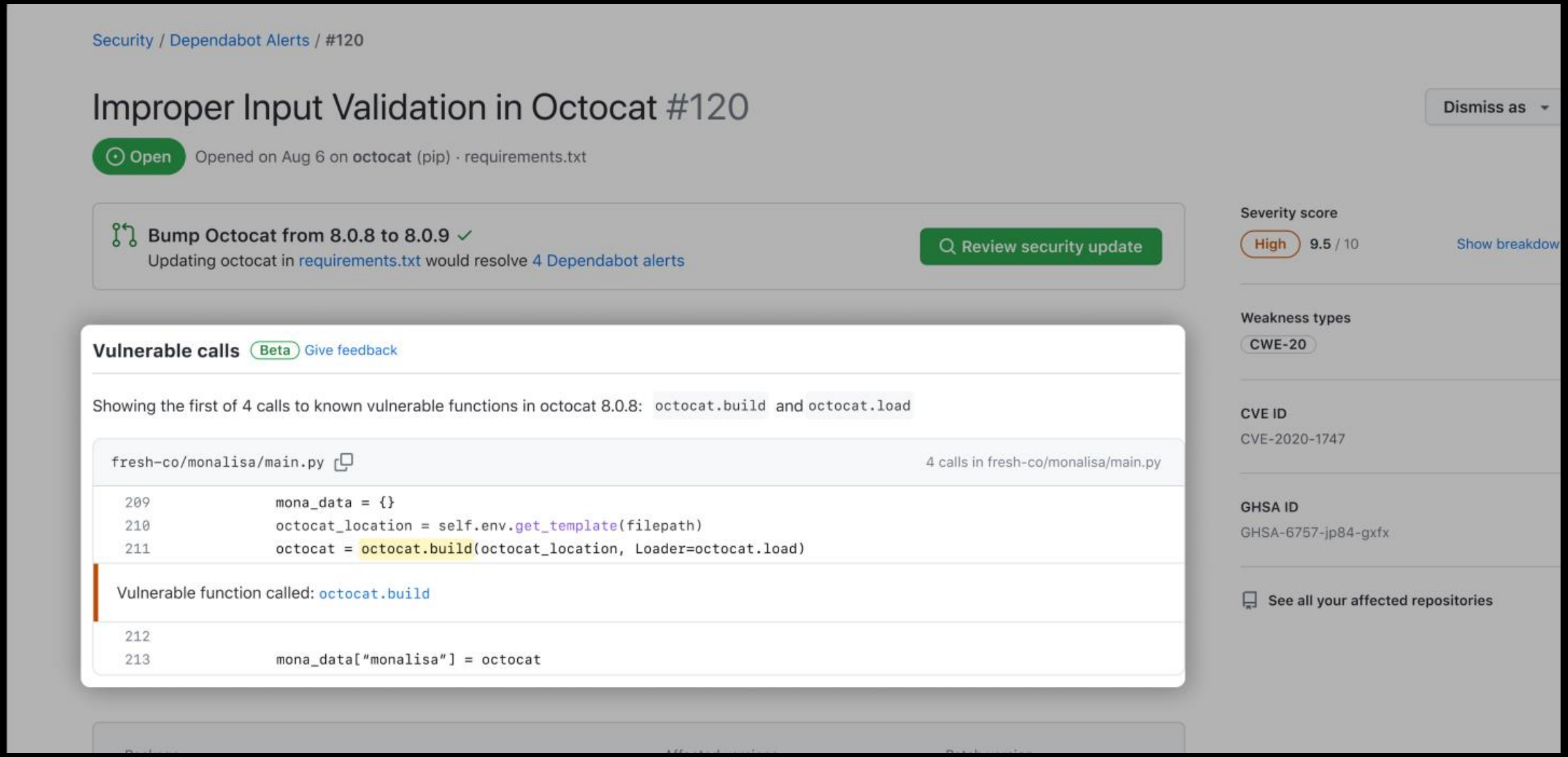
Everything is recorded; you can set granular policy on who can deploy where and for what

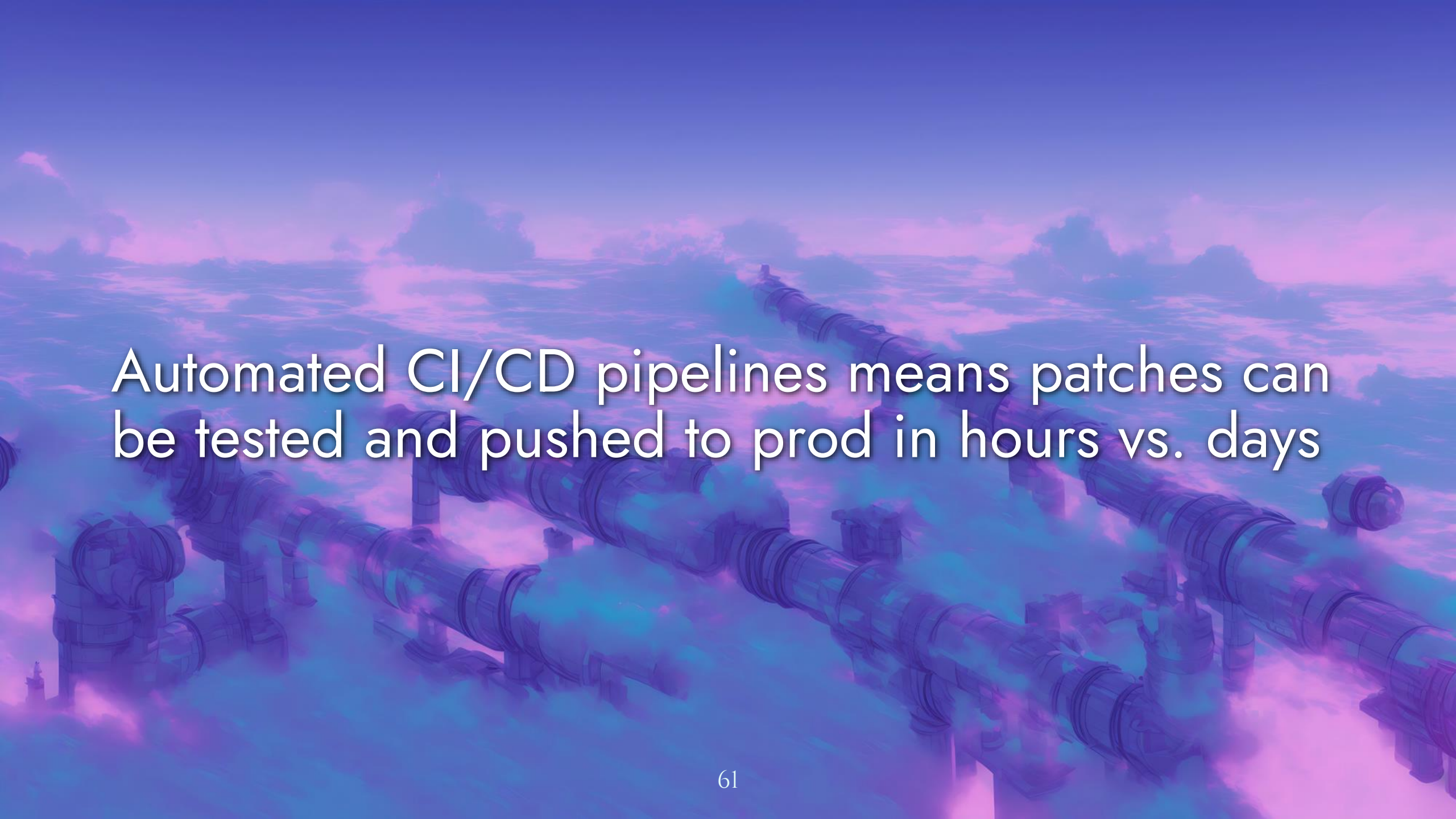
SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_


firewall-demo/banlist.txt at
shortridge-sensemaking/fire
Overview | firewall-demo | fi
Signal Sciences
https://github.com/shortridge-sensemaking/firewall-demo/blob/main/banlist.txt
shortridge-sensemaking / firewall-demo
Type / to search
Code
Pull requests 1
Actions
Security
Insights
Settings
main
firewall-demo / banlist.txt
Go to file t
swagitda Revert "Update IP Banlist"
38ead24 · 1 hour ago
History
Code
Blame
1281 lines (1280 loc) · 17.9 KB
Raw
#
#
#
Binary Defense Systems Artillery Threat Intelligence Feed and Banlist Feed
https://www.binarydefense.com
#
Note that this is for public use only.
The ATIF feed may not be used for commercial resale or in products that are charging fees for such services.
Use of these feeds for commerical (having others pay for a service) use is strictly prohibited.
#
#
#
1.10.241.225
1.85.49.110
1.183.12.102
1.206.27.29
1.215.138.43
1.235.198.19

CI/CD can help us with patching and keeping dependencies up to date

SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_

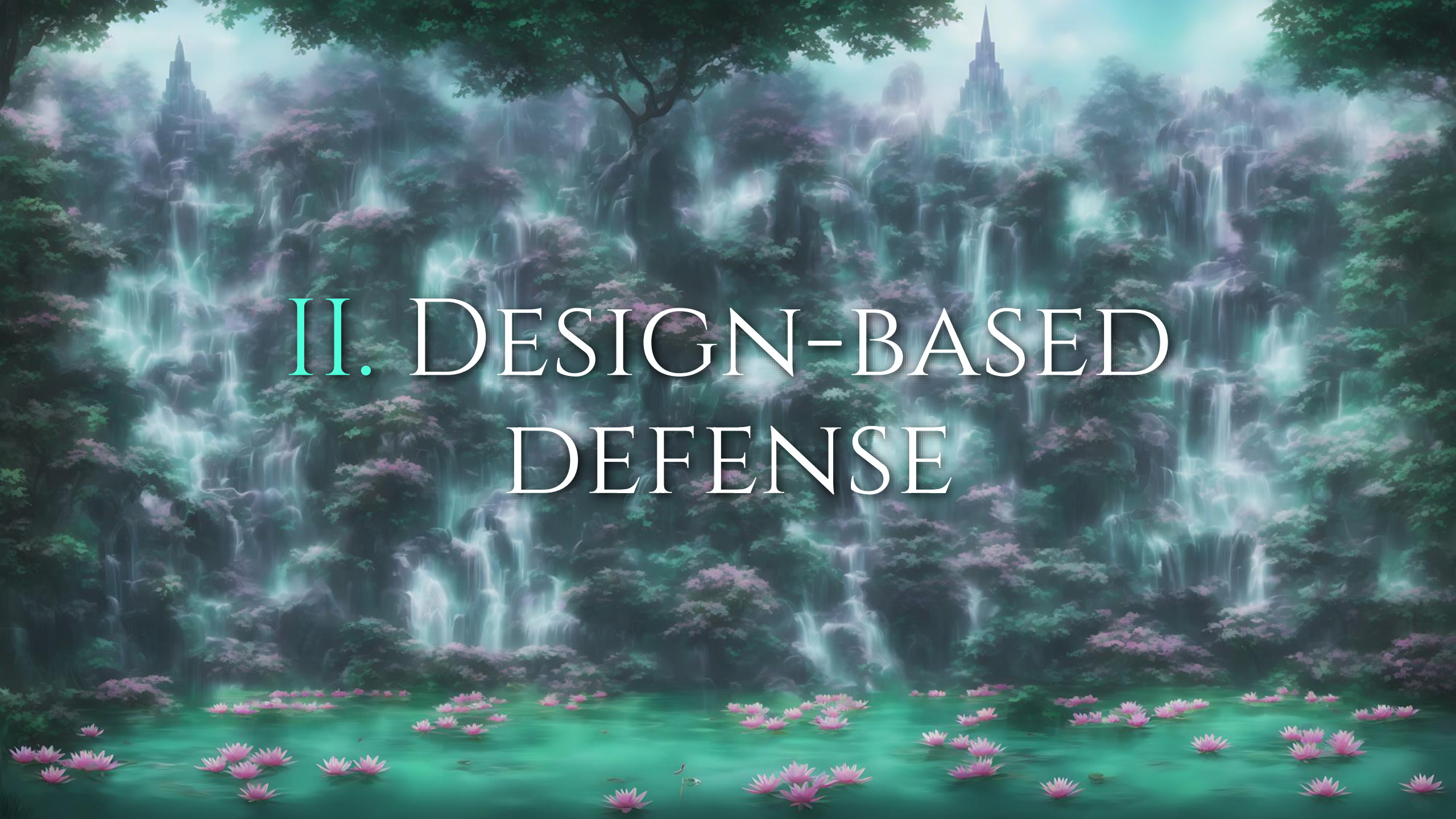

Security / Dependabot Alerts / #120
Improper Input Validation in Octocat #120
Dismiss as
Open
Opened on Aug 6 on octocat (pip) · requirements.txt
Bump Octocat from 8.0.8 to 8.0.9
Updating octocat in requirements.txt would resolve 4 Dependabot alerts
Review security update
Severity score
High 9.5 / 10
Show breakdow
Weakness types
CWE-20
CVE ID
CVE-2020-1747
GHSA ID
GHSA-6757-jp84-gxfx
See all your affected repositories
Vulnerable calls
Beta
Give feedback
Showing the first of 4 calls to known vulnerable functions in octocat 8.0.8: octocat.build and octocat.load
fresh-co/monalisa/main.py
4 calls in fresh-co/monalisa/main.py
209 mona_data = {}
210 octocat_location = self.env.get_template(filepath)
211 octocat = octocat.build(octocat_location, Loader=octocat.load)
Vulnerable function called: octocat.build
212
213 mona_data["monalisa"] = octocat


Automated CI/CD pipelines means patches can be tested and pushed to prod in hours vs. days

Update-and-patch cycles become an automatic, daily affair, freeing time for other priorities

SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_


tl;dr CI/CD lets us move faster and track the
things we do — or revert (attackers can't do so)


II. Design-based
defense

How should we prioritize the types of solutions we design? Are some better than others?

SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_


We want to design solutions that encourage the nimbleness that we envy in attackers.

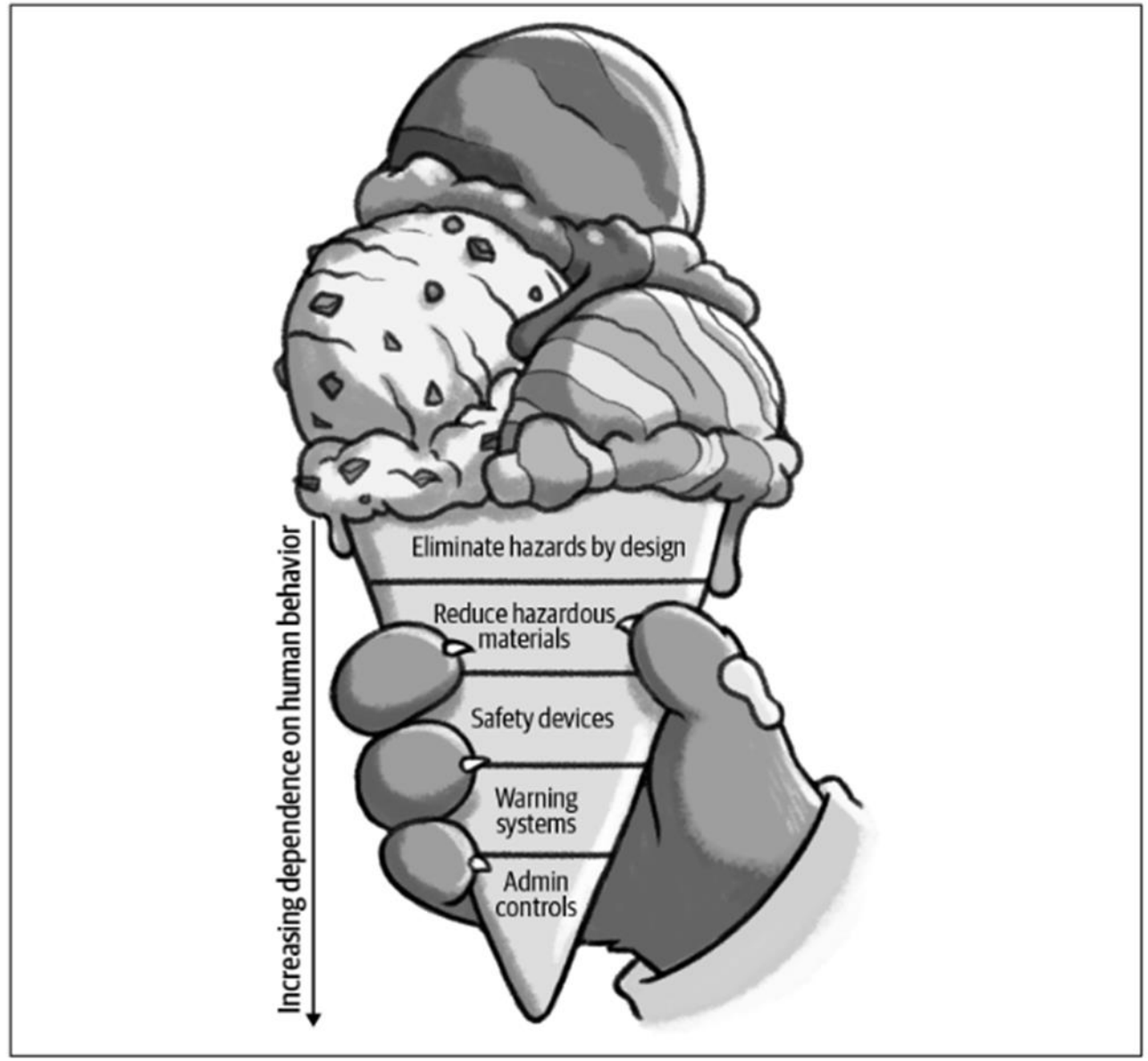
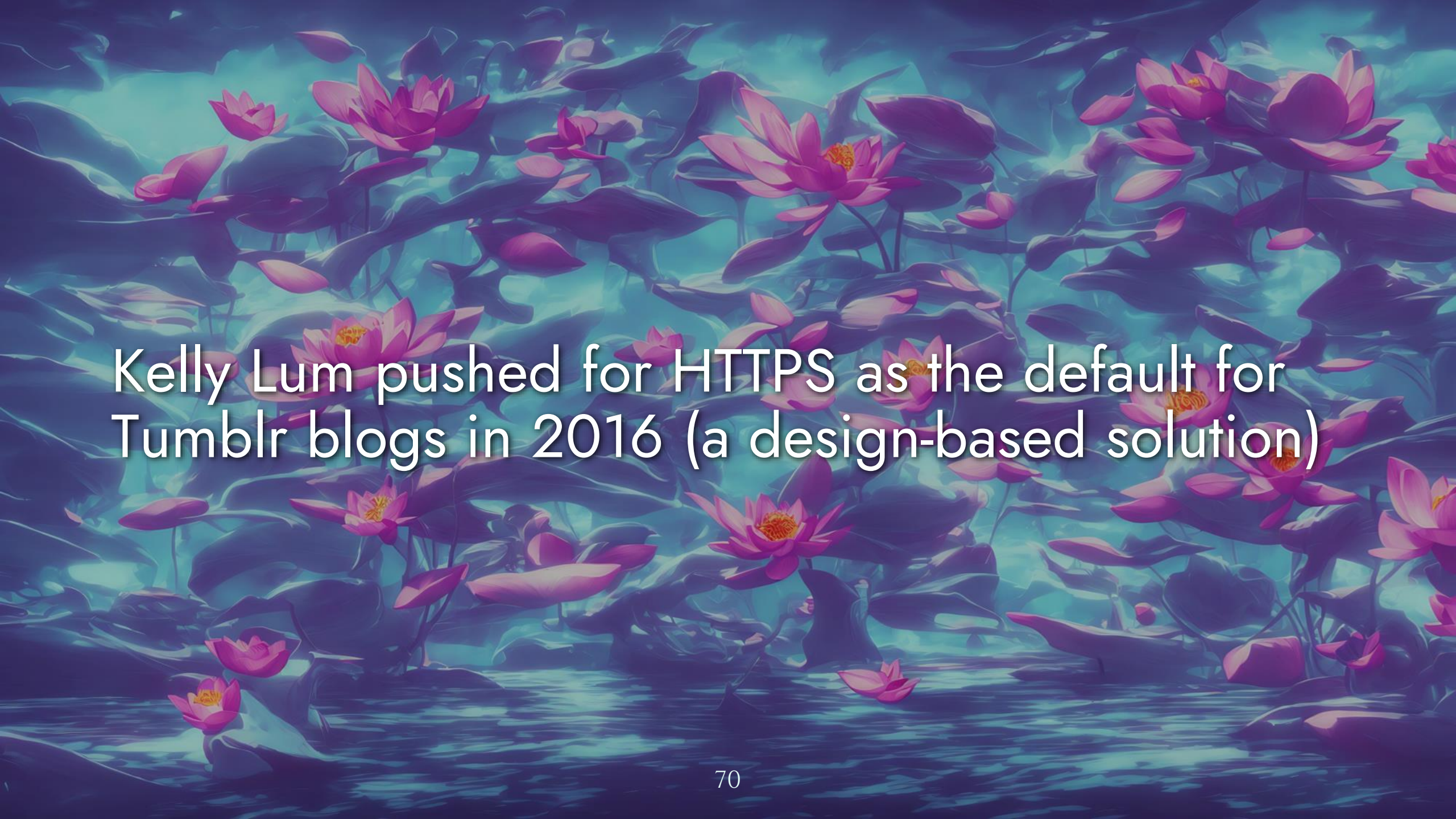


Figure 7-3. The Ice Cream Cone Hierarchy of Security Solutions


“Human fallibility is like gravity, weather, and terrain, just another foreseeable hazard.”

Finite cognitive resources; competing pressures; exhaustion, stress, distraction…

SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_


Kelly Lum pushed for HTTPS as the default for Tumblr blogs in 2016 (a design-based solution)

Isolation, standardization, message buses, declarative dependencies, queues, failover…

SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_


Modularity

Modularity: allows structurally or functionally distinct parts to retain autonomy during periods of stress & allows for easier recovery from loss

SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_


Unless components can fail independently, you don’t have modularity in the resilience sense.

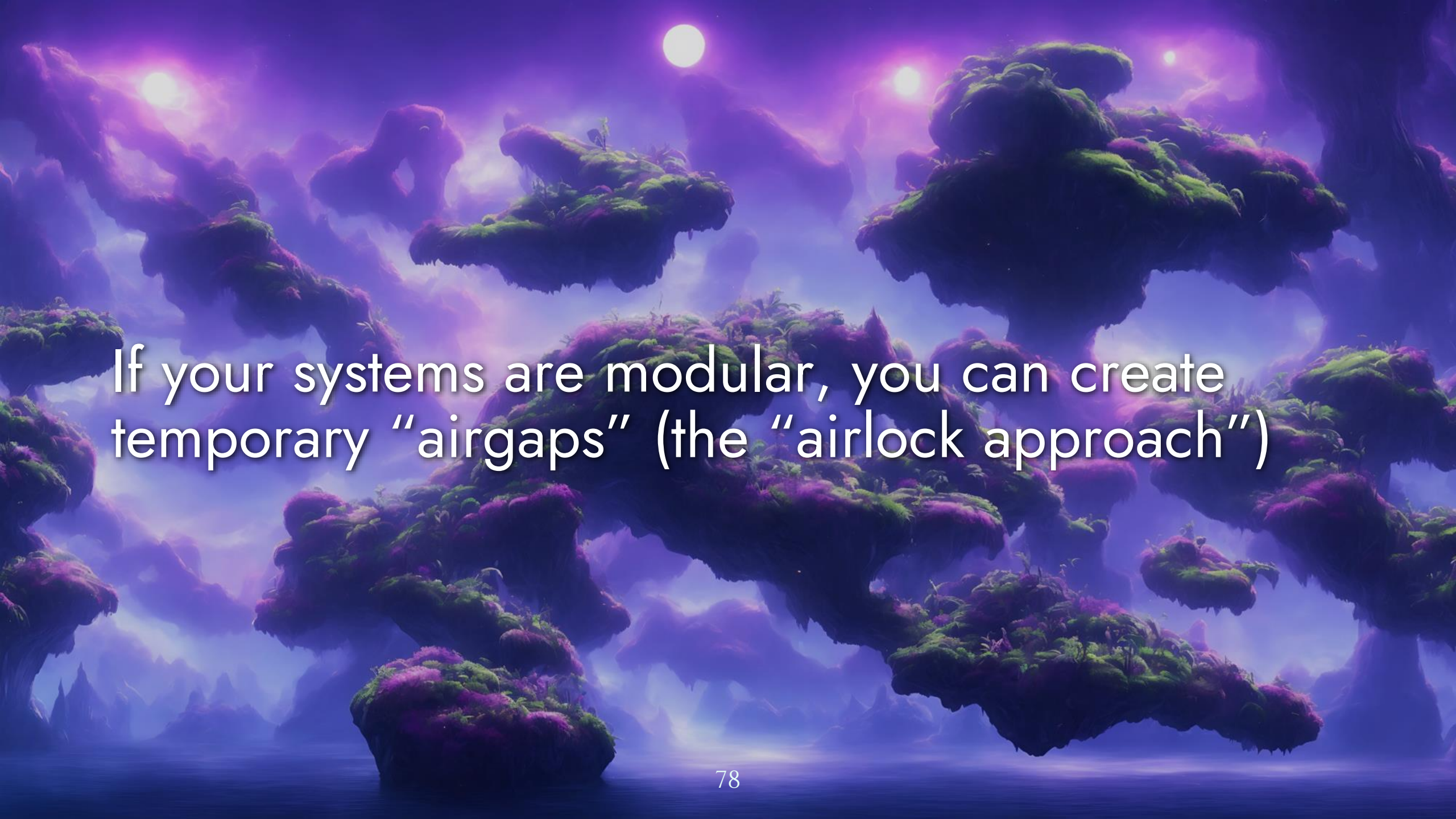
Queues and message brokers support modularity, each in different ways...

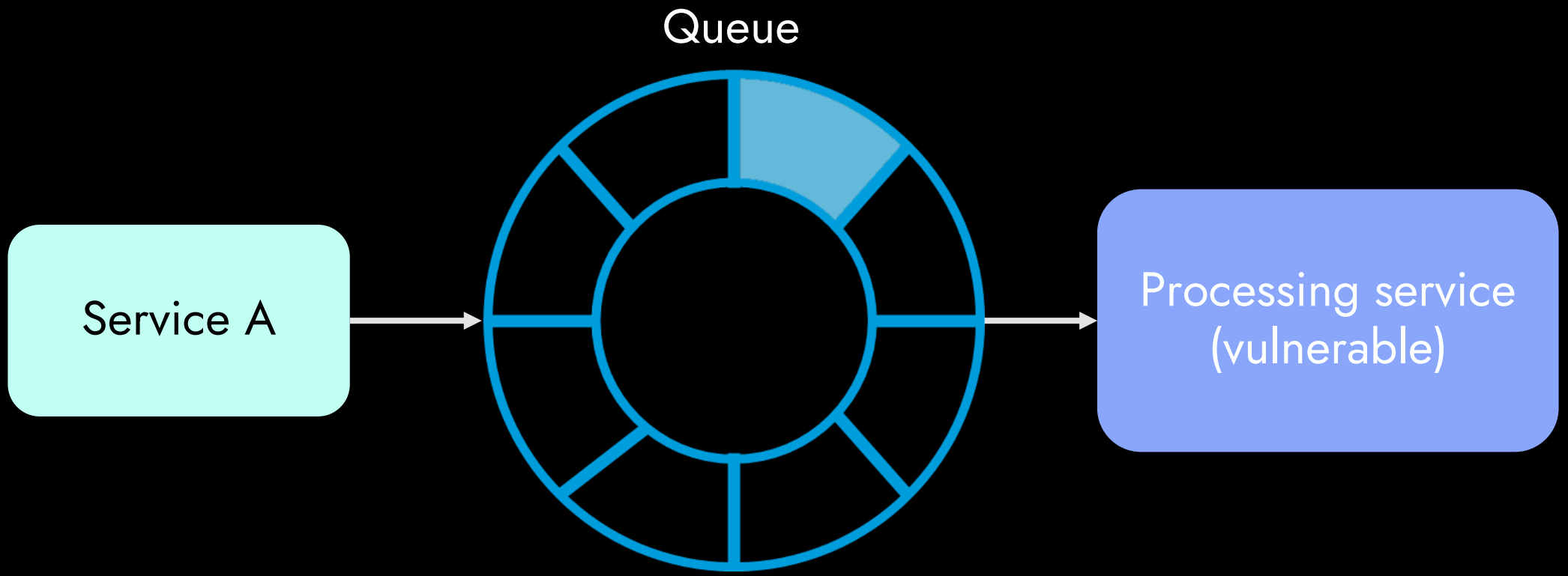
SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_


A queue adds a buffer; a message broker can replay and make return code non-blocking.

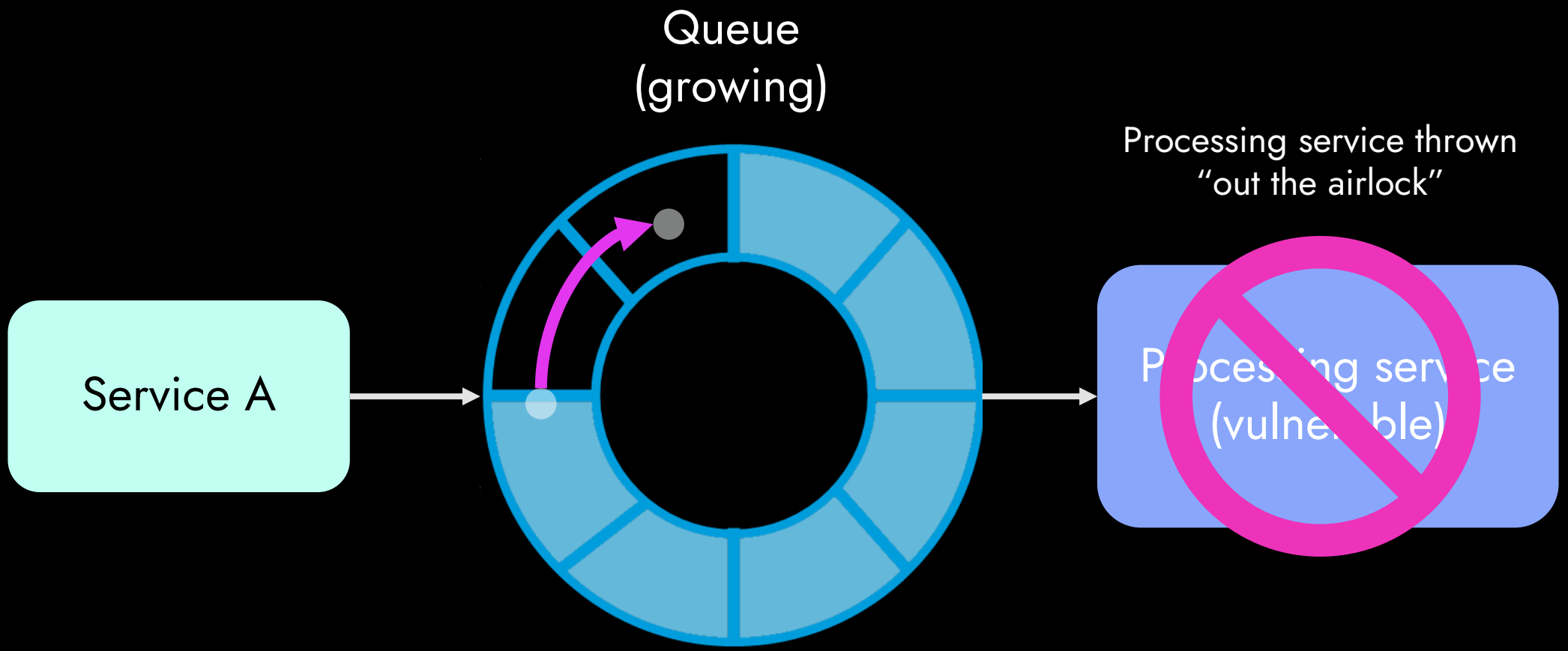
Both tools standardize how services pass data around and provide a centralized view.

SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_

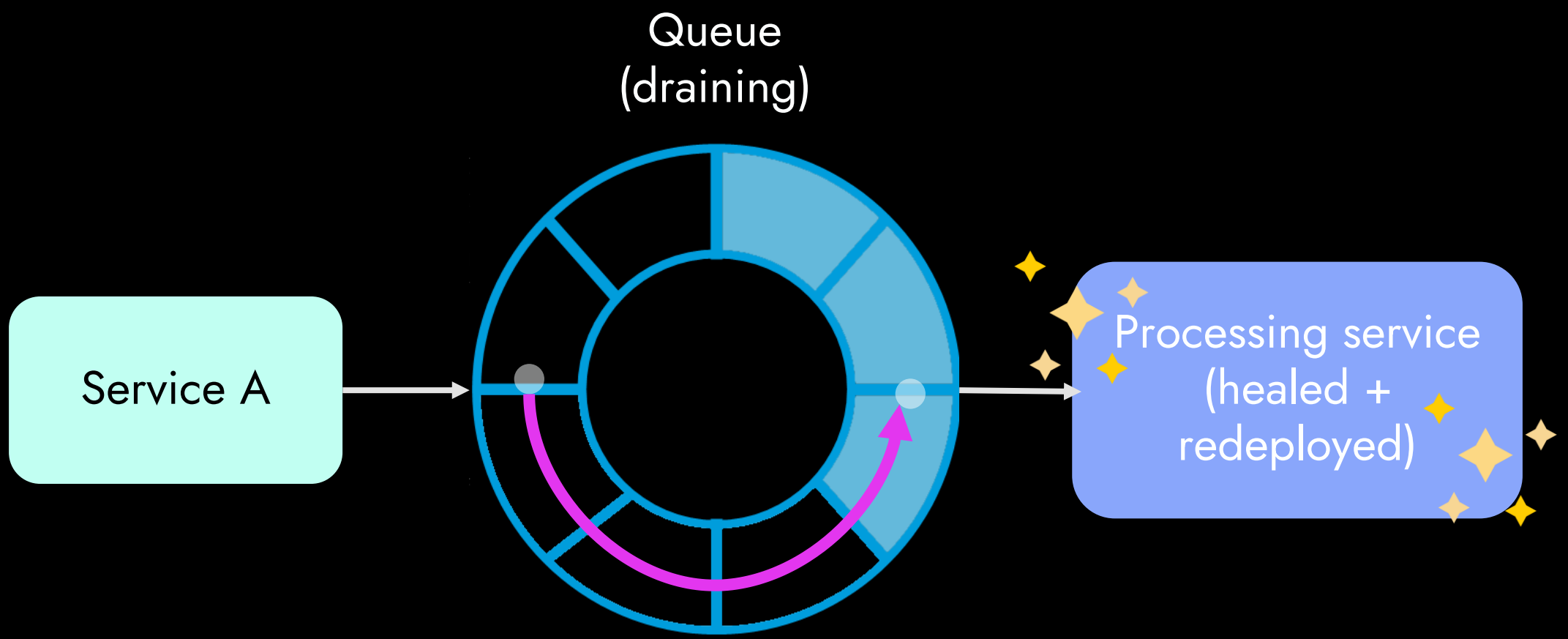

If your systems are modular, you can create temporary “airgaps” (the “airlock approach”)



SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_



SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_



SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_


Modularity minimizes incident impact – think ransomware in serverless (it doesn't happen)

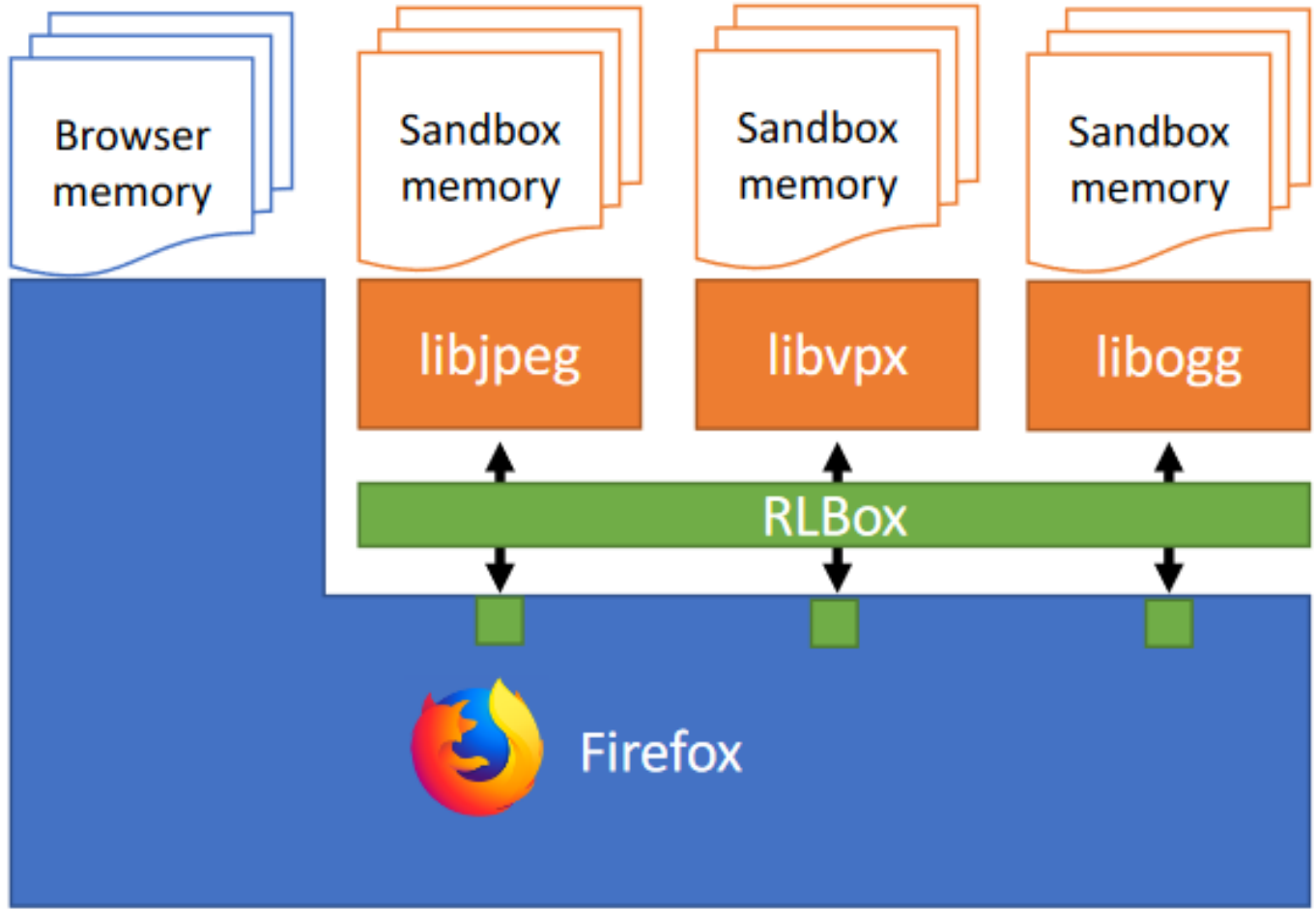
Modularity allows for basic encapsulation and separation of concerns... and supports isolation

SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_

Here's what it's like to live in 2023 with a strong engineering culture:

RLBox: trap C code in a WebAssembly (Wasm) sandbox to isolate hazardous subcomponents

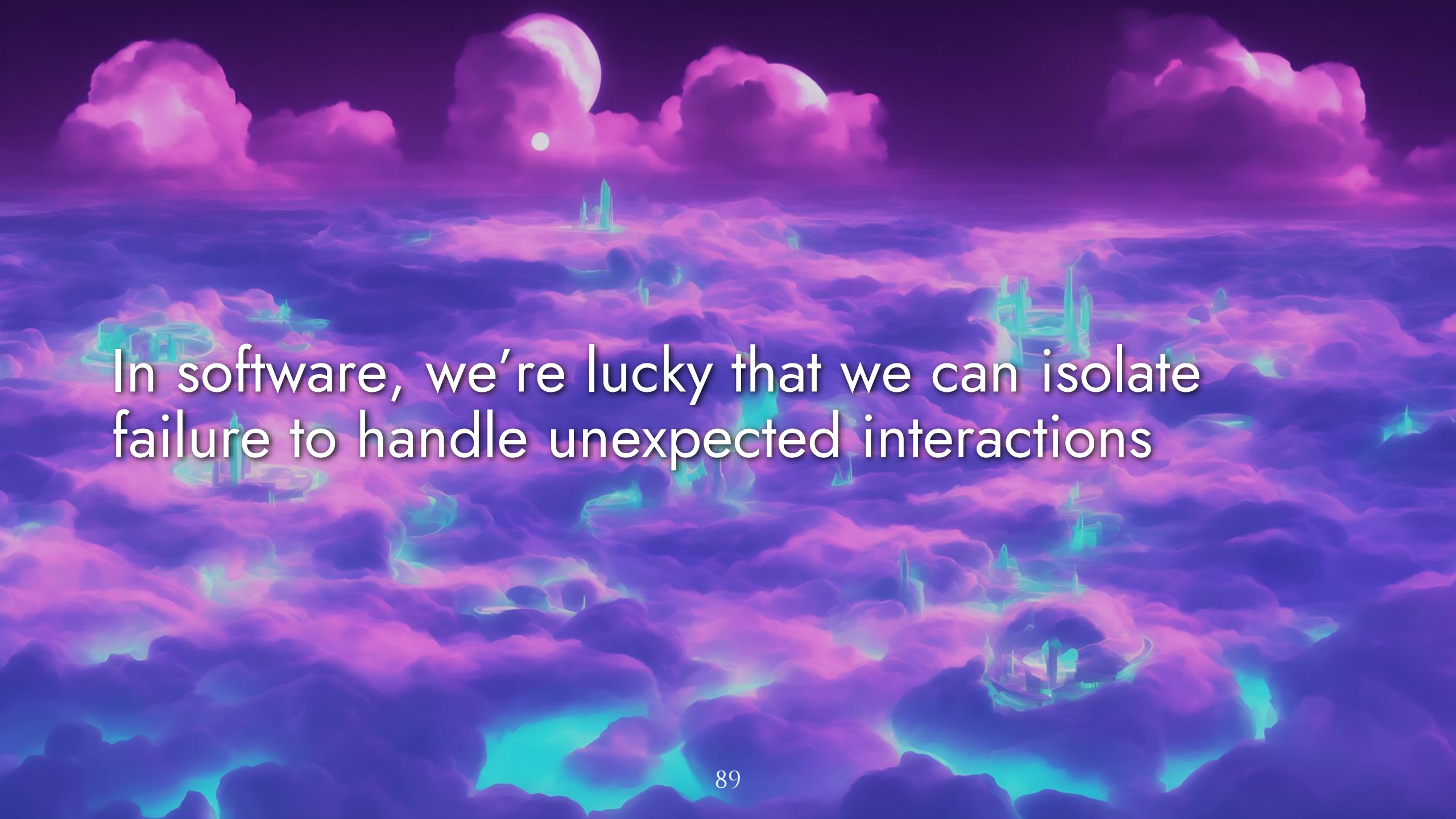
SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_


Browser memory
Sandbox memory
Sandbox memory
Sandbox memory
libjpeg
libvpx
libogg
RLBox
Firefox


Imagine not worrying about 0day anymore*.

You've been so focused on AI you've missed groundbreaking things like this. It's sad. :(

SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_

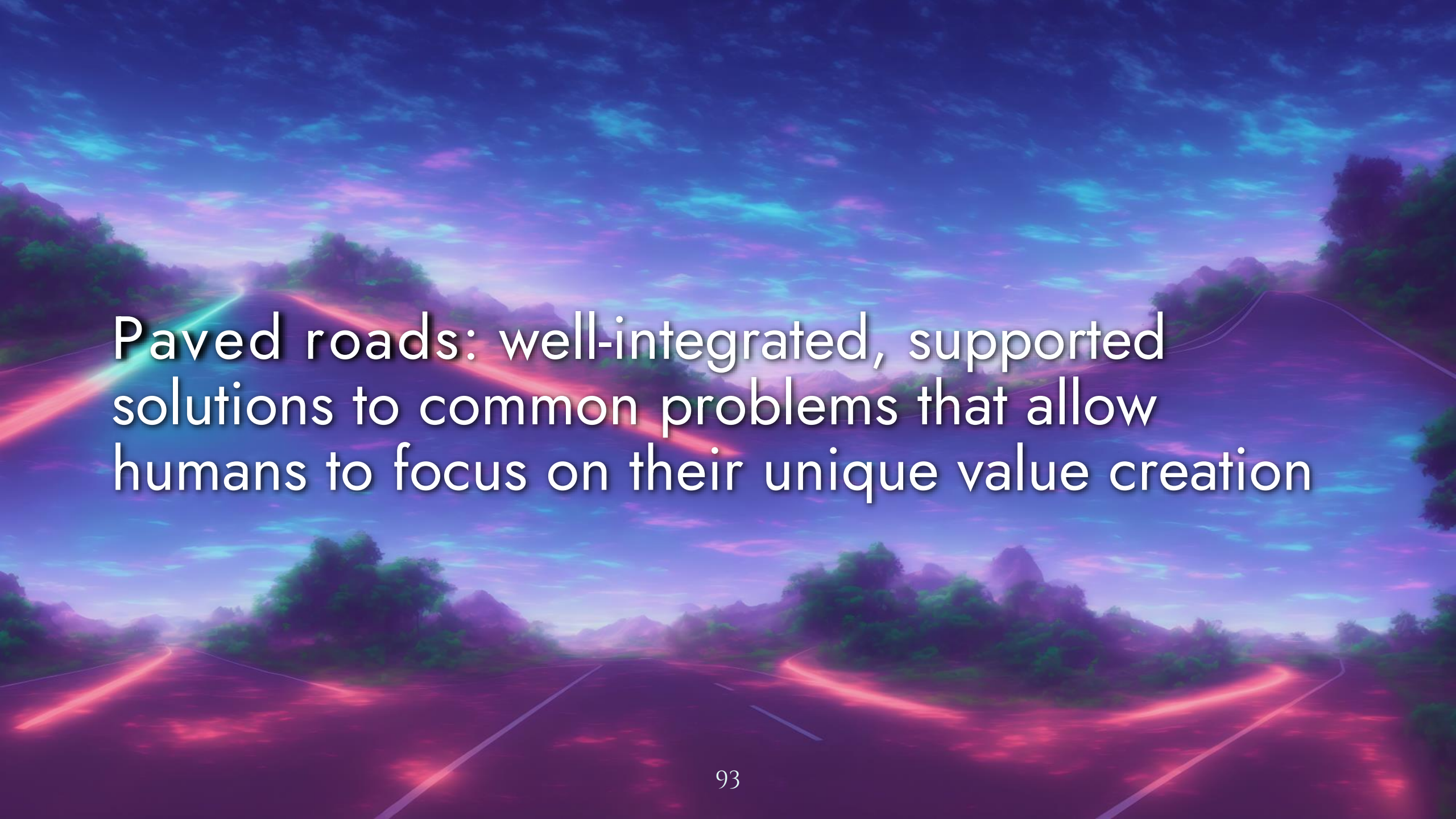

In software, we're lucky that we can isolate
failure to handle unexpected interactions

Start “boring”: set AWS security groups – or use serverless functions, containers, or VMs

SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_


If a vulnerable component is in a sandbox, the attacker faces a challenge to reach their goal

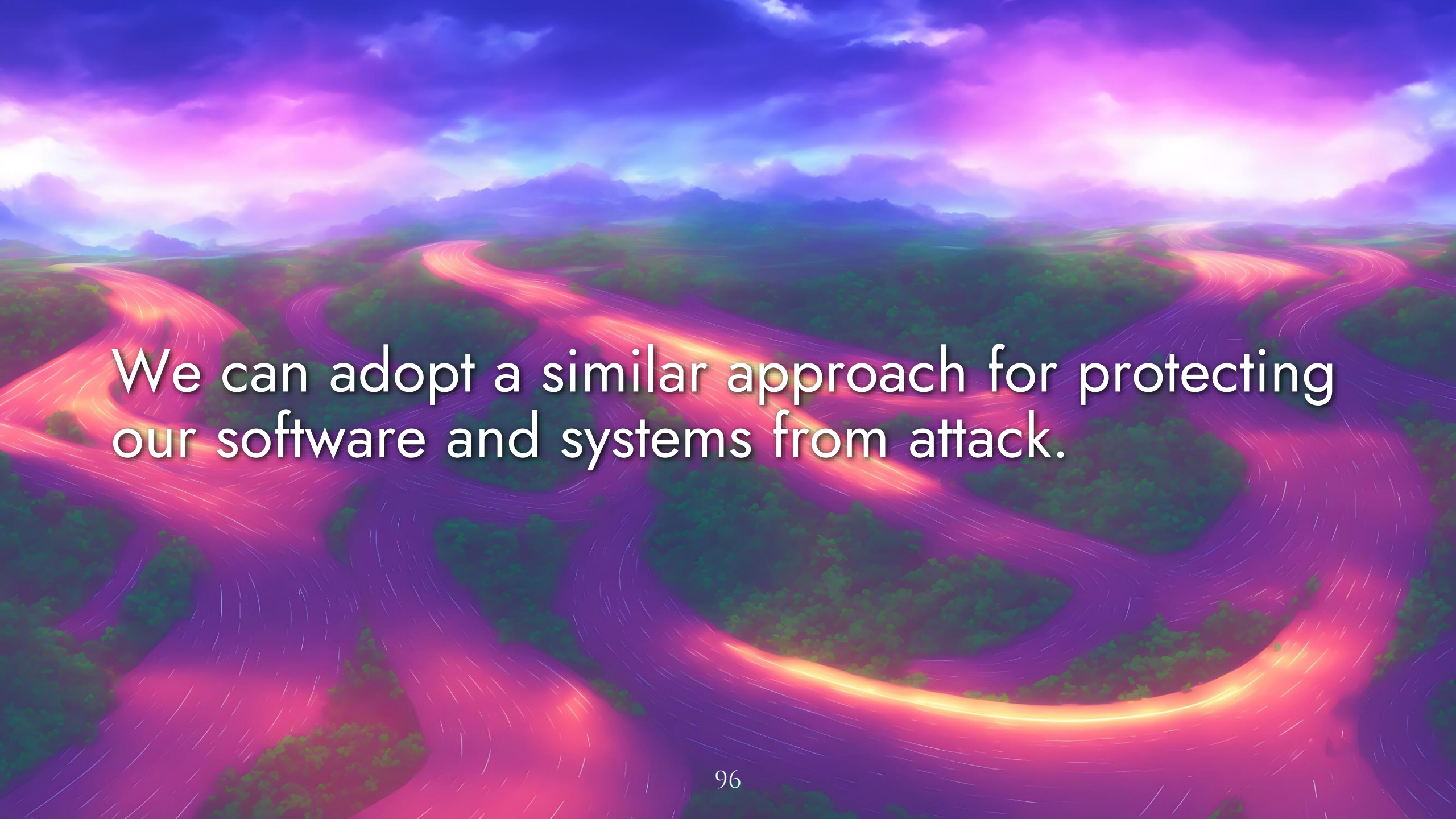

Paved Roads


Paved roads: well-integrated, supported solutions to common problems that allow humans to focus on their unique value creation

Attackers have paved roads, like Cobalt Strike – it makes the easy way the pwnful way.

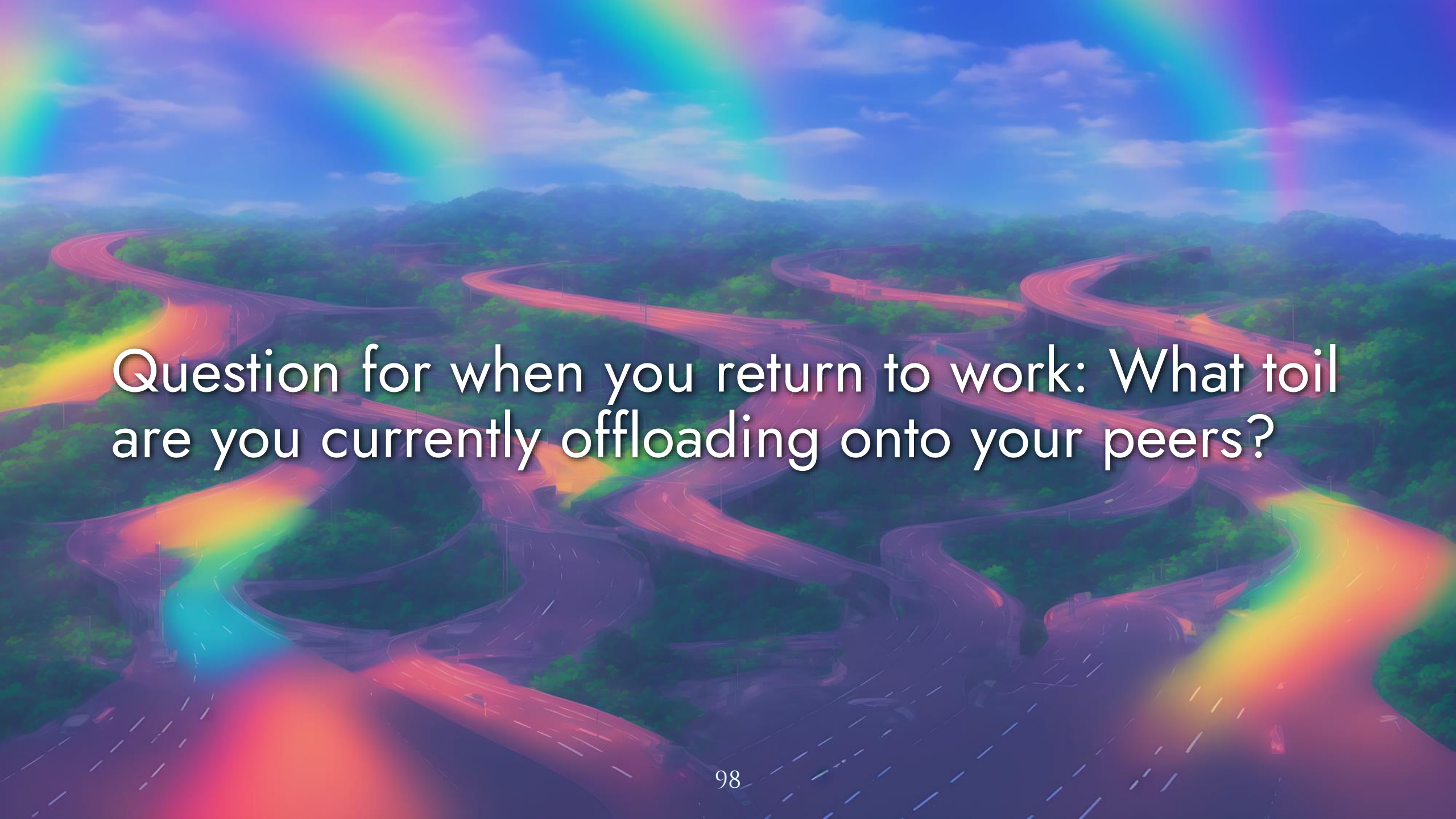
SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_

Hyperscale nation states love building platforms and toolchains for their attack ops, too


We can adopt a similar approach for protecting our software and systems from attack.

Netflix: Wall-E framework turns security requirements into filters to replace checklists

SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_


Question for when you return to work: What toil are you currently offloading onto your peers?

“The bulk of the ‘going internet-facing’ checklist boiled down to one item: Will you use Wall-E?”

SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_

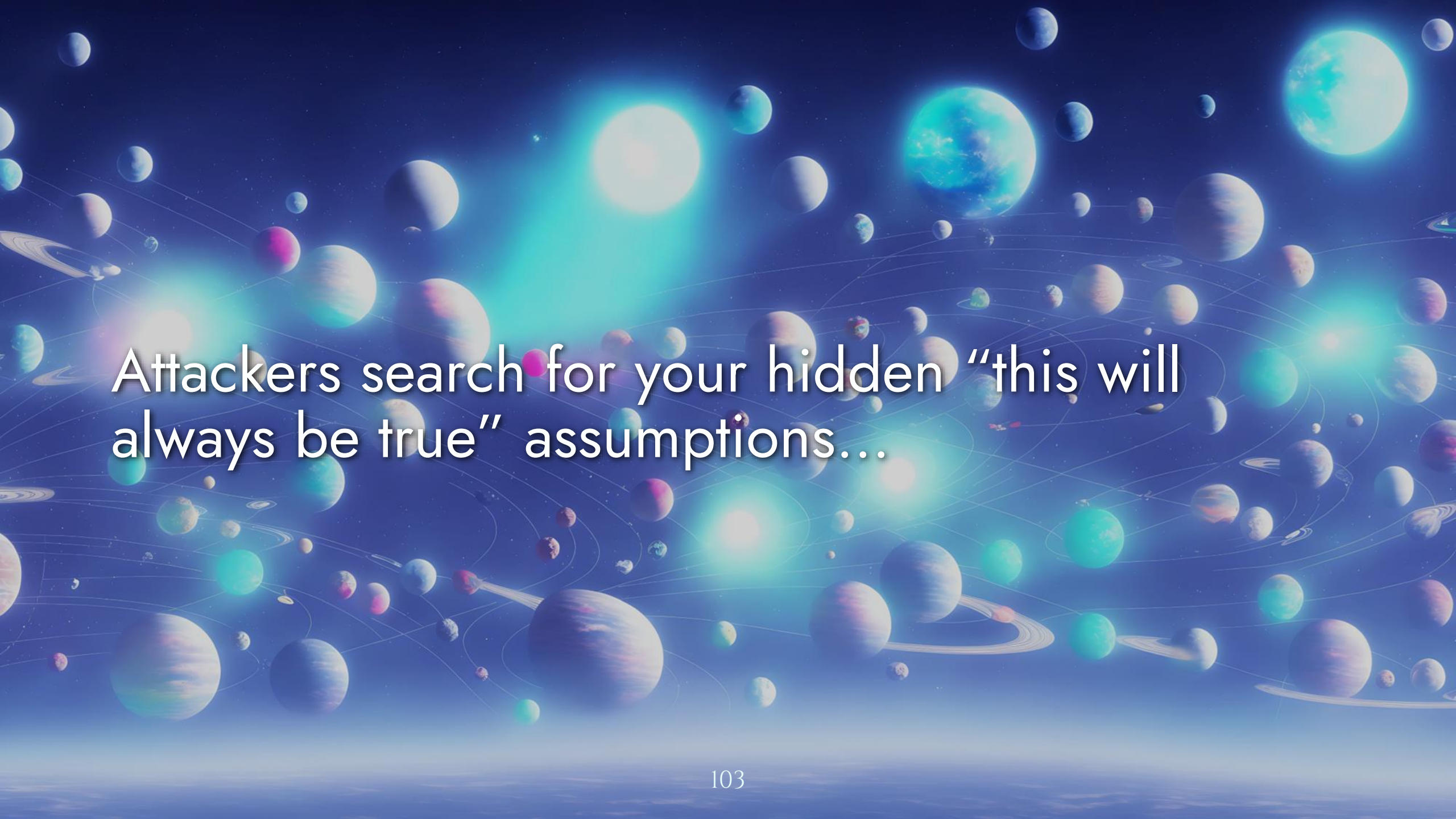
Block: enabling backend services to securely connect across business unit boundaries

SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_


III. Systems Thinking

Attackers think in systems while defenders think in components. It doesn't have to be this way.

SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_


Attackers search for your hidden “this will always be true” assumptions…

Then they ask, “you say this will always be true; is that the case?” to break those assumptions

SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_

Attackers target our “this will always be true” assumptions that exist all over our stack.

SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_

Parsing this string will always be fast

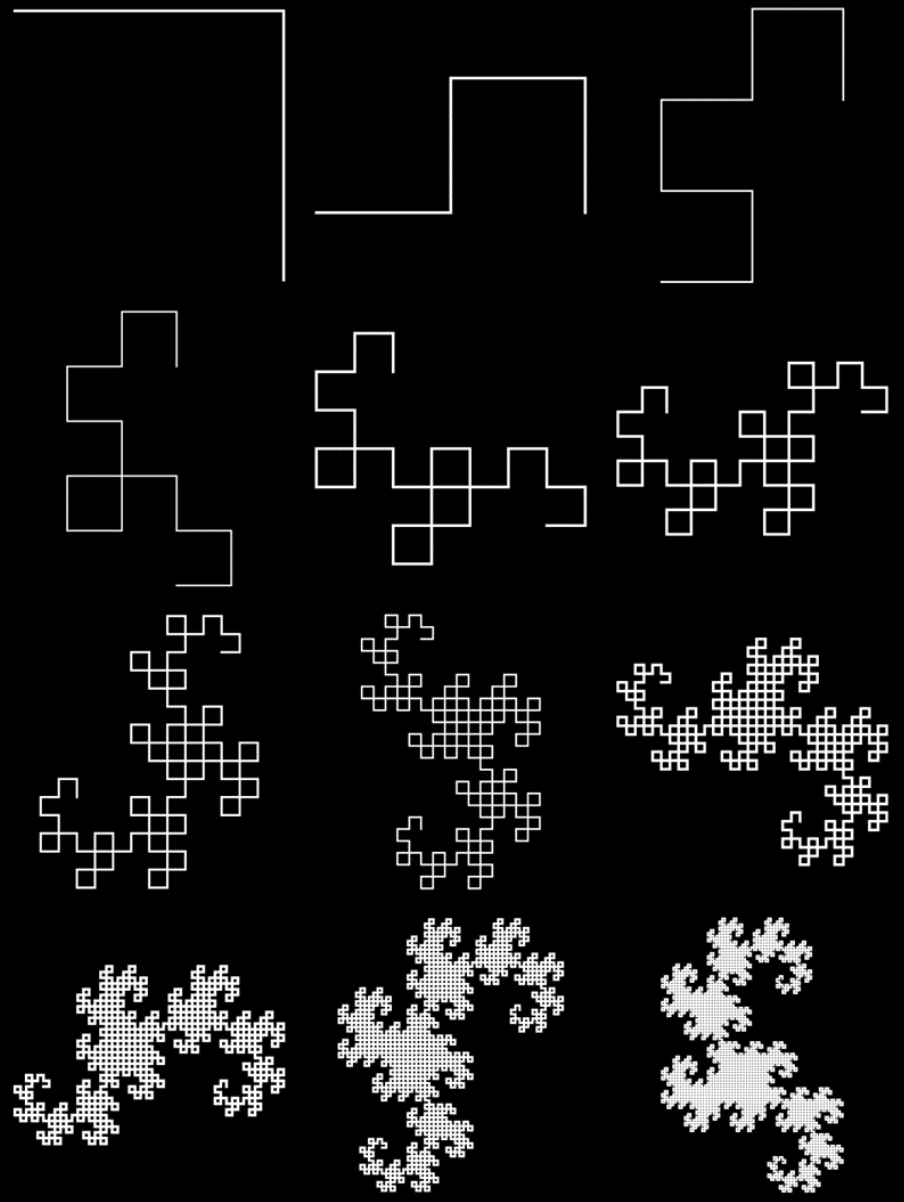

Messages on this port will always be post-auth


An alert will always fire if a malicious executable appears

The attacker thinks, “They say X here, but I can show that it isn’t quite true… interesting. Let’s keep looking to see if they’re just a little wrong or *really* wrong.”

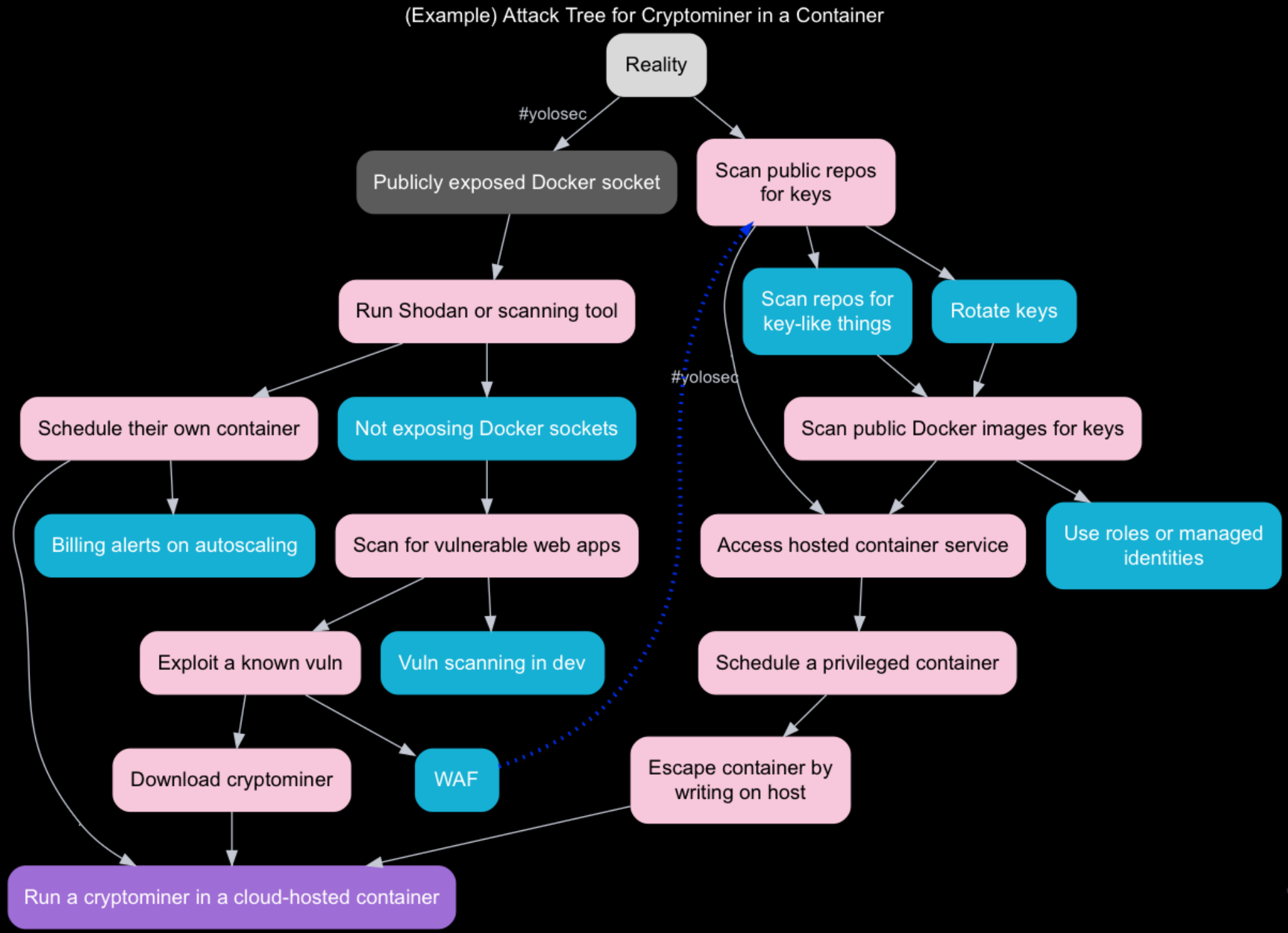
SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_

We can adopt a similar process through decision trees and resilience stress testing



SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_


We can refine our mental models continuously
rather than waiting for attackers to exploit them


(Example) Attack Tree for Cryptominer in a Container
Reality
#yolosec
Publicly exposed Docker socket
Scan public repos for keys
Run Shodan or scanning tool
Scan repos for key-like things
Rotate keys
#yolosec
Schedule their own container
Not exposing Docker sockets
Scan public Docker images for keys
Billing alerts on autoscaling
Scan for vulnerable web apps
Access hosted container service
Use roles or managed identities
Exploit a known vuln
Vuln scanning in dev
Schedule a privileged container
Download cryptominer
WAF
Escape container by writing on host
Run a cryptominer in a cloud-hosted container

Resilience stress tests help us identify the confluence of conditions where failure happens

How do disruptions impact the entire system's ability to recover and adapt?

SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_

We can move fast and observe how failure unfolds in our systems through experiments

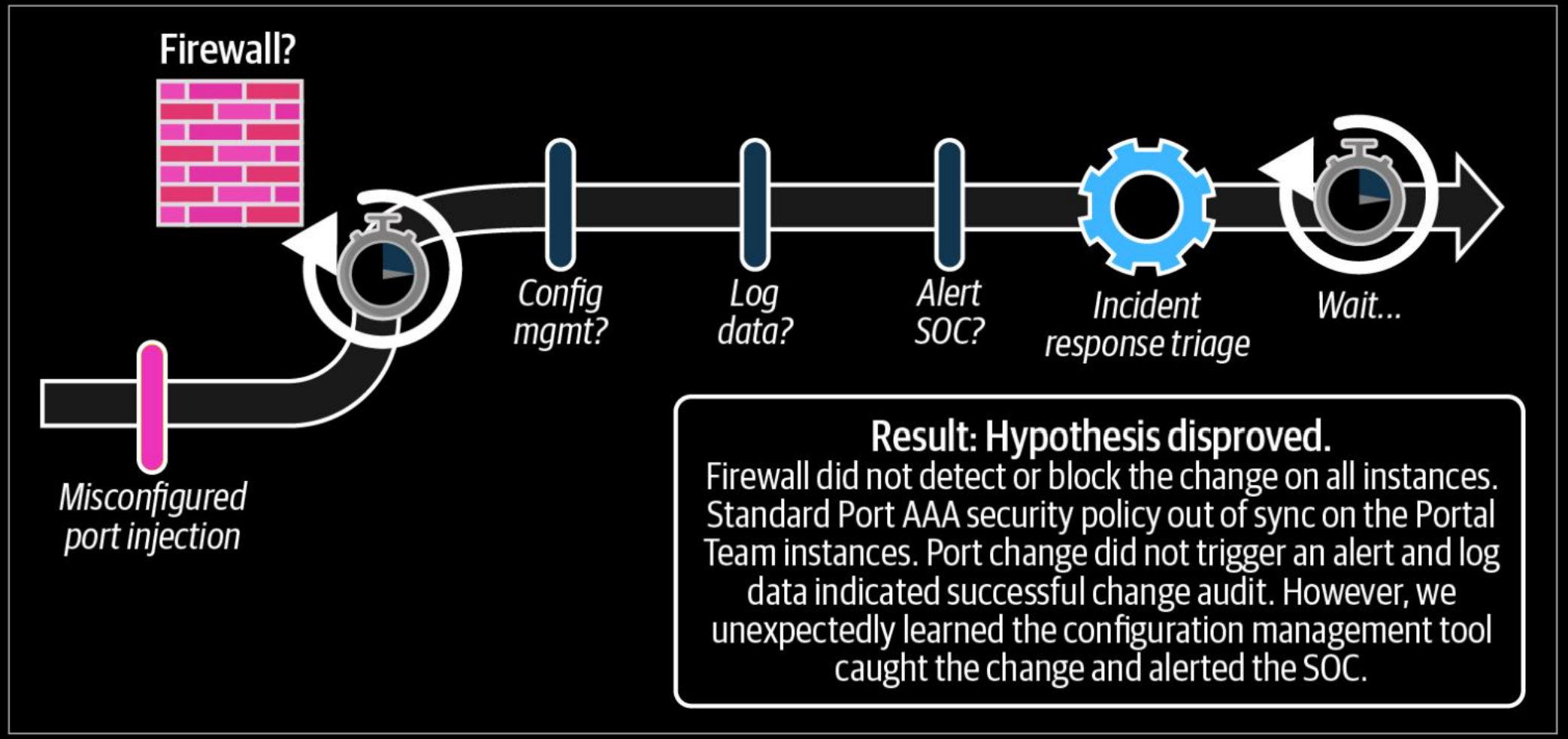


Figure 2-6. An example security chaos experiment simulating a misconfigured port injection scenario

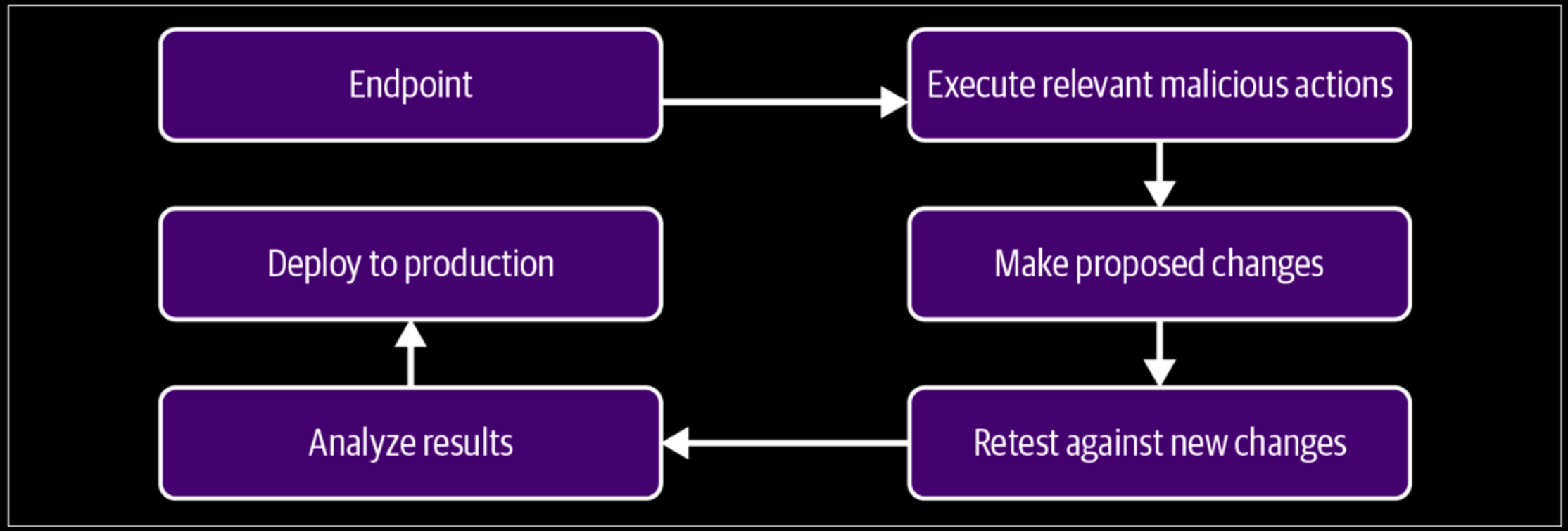


Figure 9-5. Engineering workflow change evaluation

Verizon: deploy a pod containing known vulns on a target cluster to test security controls

SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_

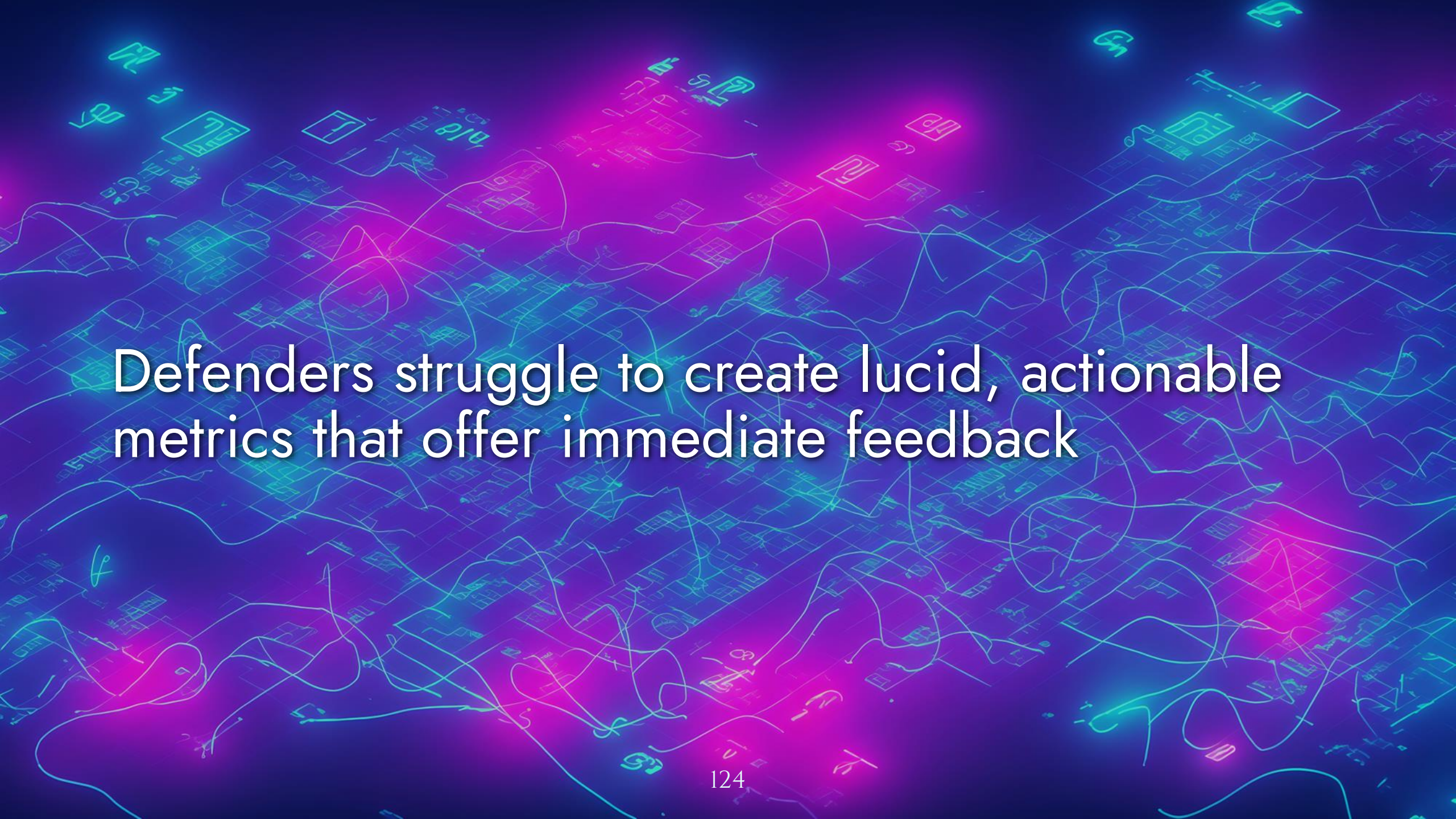
If we adopt this across the industry, vendors now must prove their products work... >:)

IV. Tangible Success

Attackers can measure tangible success and receive immediate feedback on their metrics

Do they have access, how much access do they have, and have they accomplished their goals?

SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_


Defenders struggle to create lucid, actionable metrics that offer immediate feedback

CISOs, your “risk coverage” and “time to detect” mean nothing, it’s embarrassing

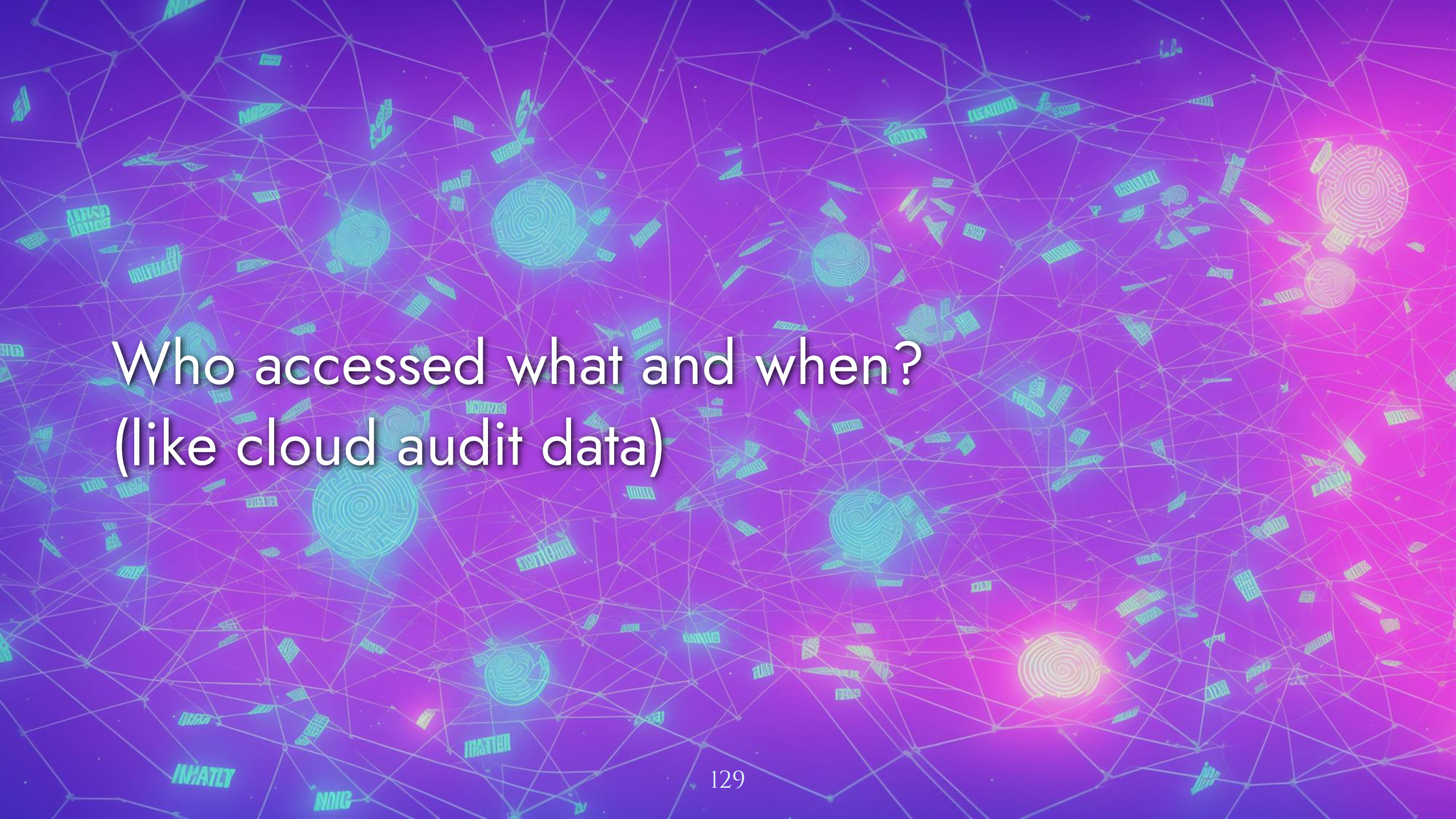
SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_


System signals

Reliability signals also benefit systems security

SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_

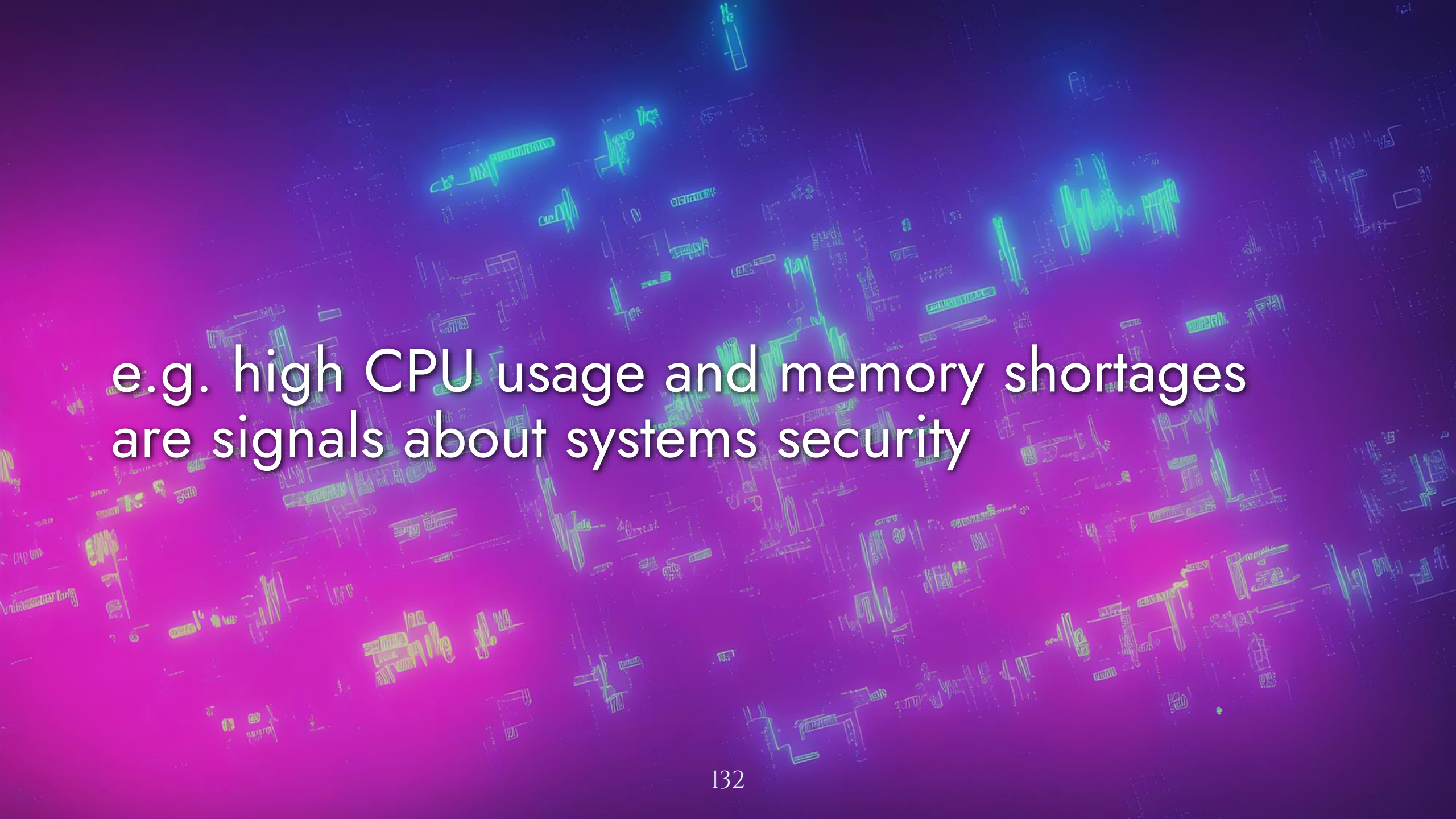

Who deployed what and when?
(like orchestrator and deployment logs)


Who accessed what and when?
(like cloud audit data)


Database logs, billing records, netflow,
production crash dumps, error messages...

Traditional infosec doesn't measure load, latency, performance, or throughput (a mistake)

SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_


e.g. high CPU usage and memory shortages
are signals about systems security

Well-resourced attackers will monitor the system they're attacking to avoid hitting limits or alarms

SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_


Experiments can reveal what signals you *should* be collecting – don't take visibility for granted

So, what system signals can indicate attacks?
Turns out SREs and DevOps are our bffs...

SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_

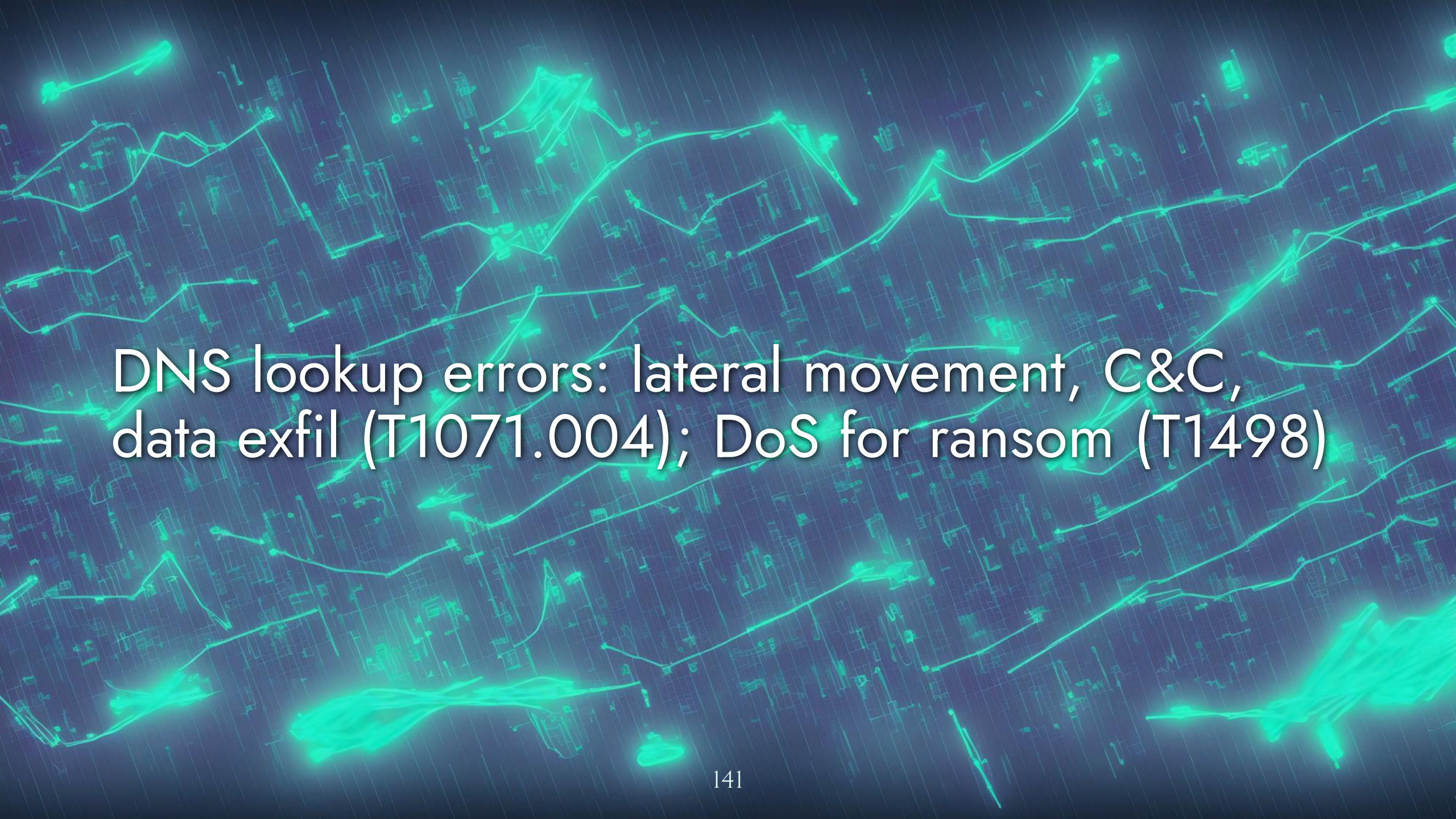
Accept queue depth: attacker hijacking system execution (T1574) or process hollowing (T1055)


Autoscale replica count: lateral movement (T1072); cryptomining; brute forcing (T1110)

Billing alerts: cloud priv escalation (T1078); crypto-mining (T610); querying data for recon

Cache hit rate (CHR): DoS; data exfiltration (T1567); brute forcing


Disk usage, throughput, & IOPS: ransomware (T1486); staging data for exfiltration (T1074)


DNS lookup errors: lateral movement, C&C,
data exfil (T1071.004); DoS for ransom (T1498)

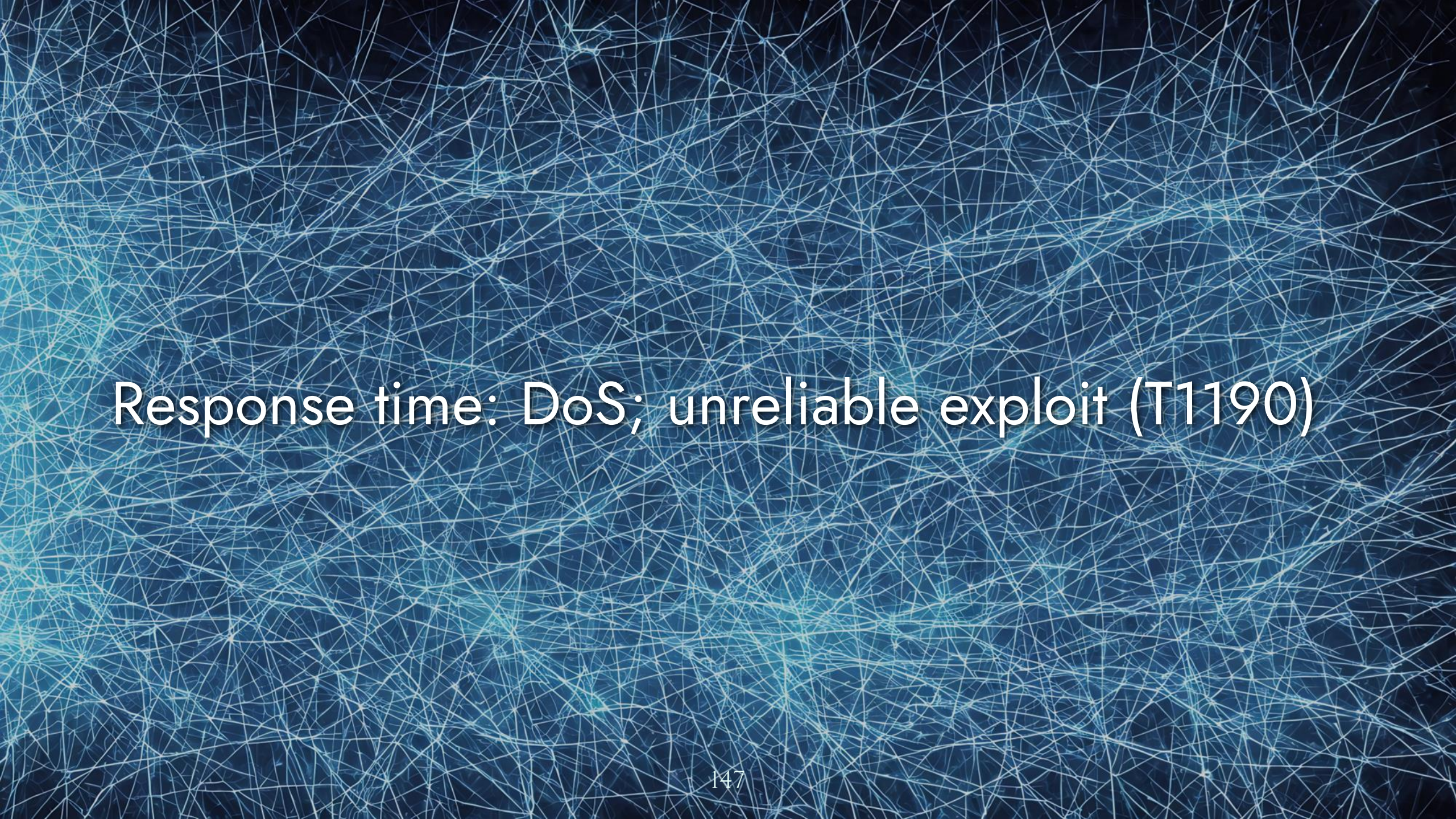
Error rate: credential stuffing (T1110) or DoS

Heartbeat response: endpoint DoS (T1499); restricting connections for evasion (T1562)

Rate limit availability: SSRF (T1190); brute force logins (T1110)

Replication lag: unauthorized access or modification (T1565); exploiting inconsistencies

Resource consumption creeping towards max levels (CPU, memory): cryptominers; hijacking resources (T1496); in-memory attacks (T1055)


Response time: DoS; unreliable exploit (T1190)

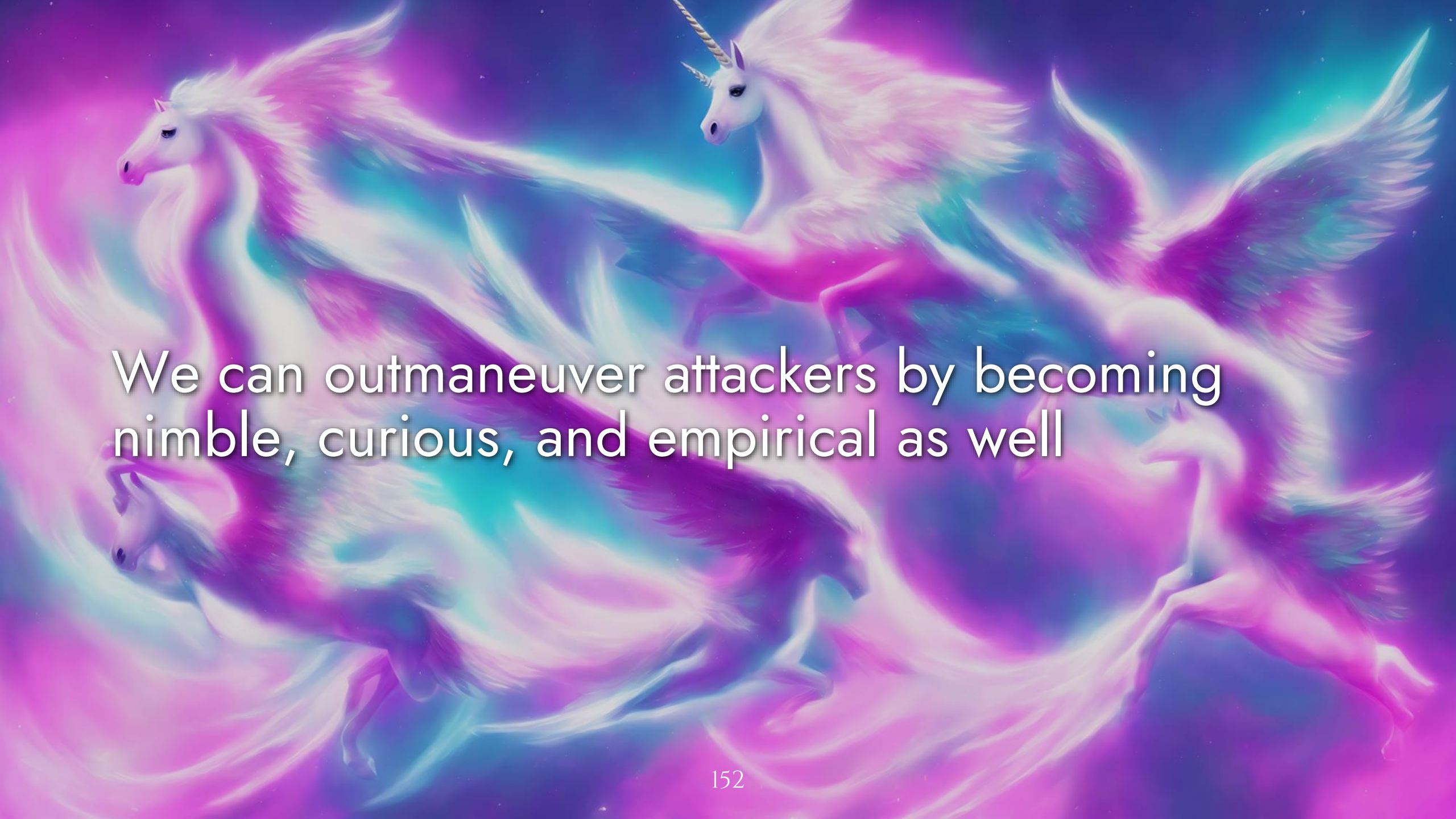
Swap usage: data exfiltration (T1074.001)

System log lag: stopping or deleting logs to conceal attack operations (T1070)

We need our feedback loops to give us immediate sensory input like attackers get

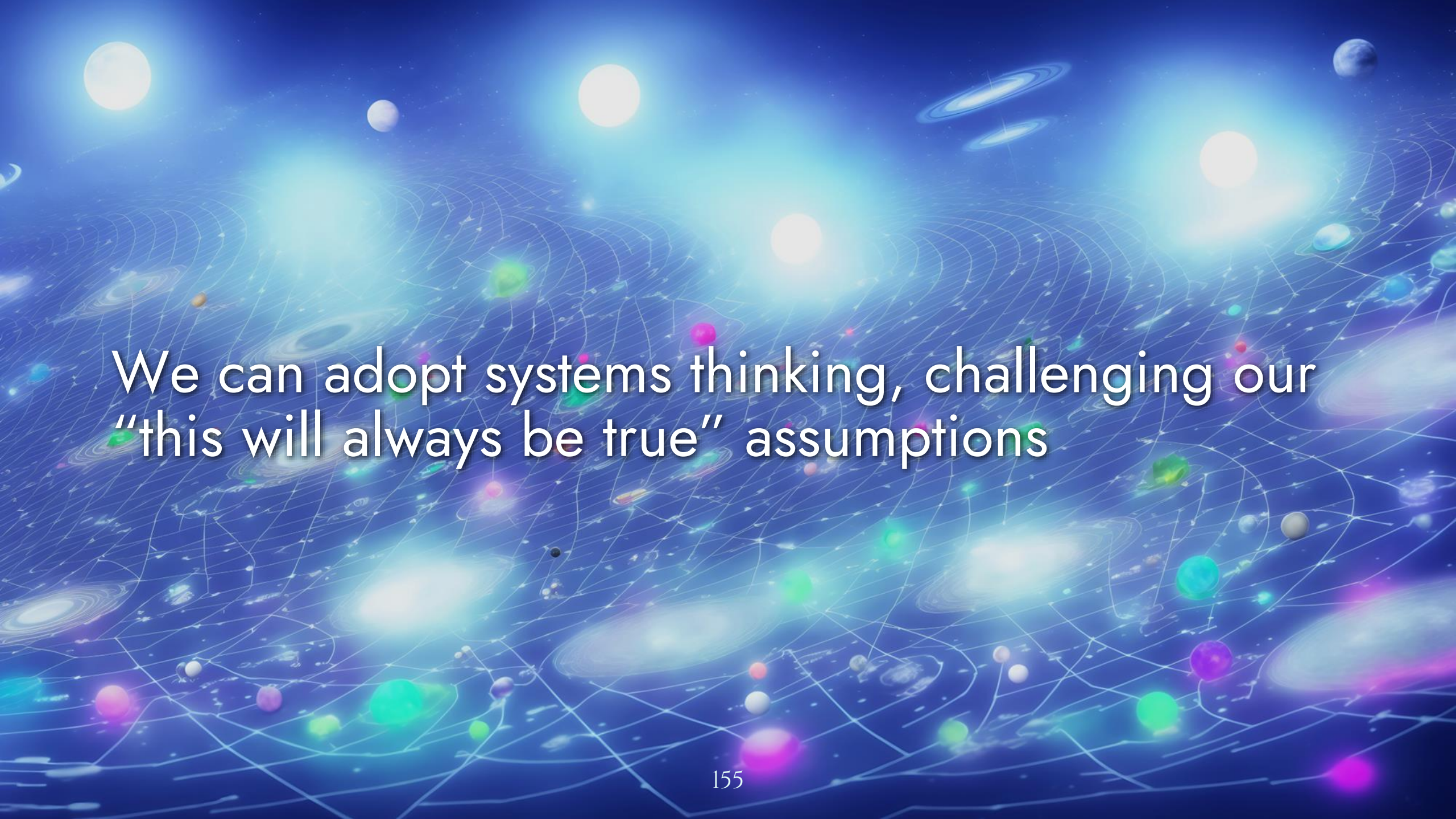
SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_


Viva ~~Las Vegas~~
la Révolution


We can outmaneuver attackers by becoming nimble, curious, and empirical as well

We can adopt a faster tempo via Configuration as Code (CaC) and automation like CI/CD


We can pursue design-based solutions with our Ice Cream Cone Hierarchy and Paved Roads


We can adopt systems thinking, challenging our “this will always be true” assumptions

We can cultivate tangible success outcomes that leverage system signals for immediate feedback


We can fuel a feedback loop to gracefully
respond to attacks and adapt as attackers evolve

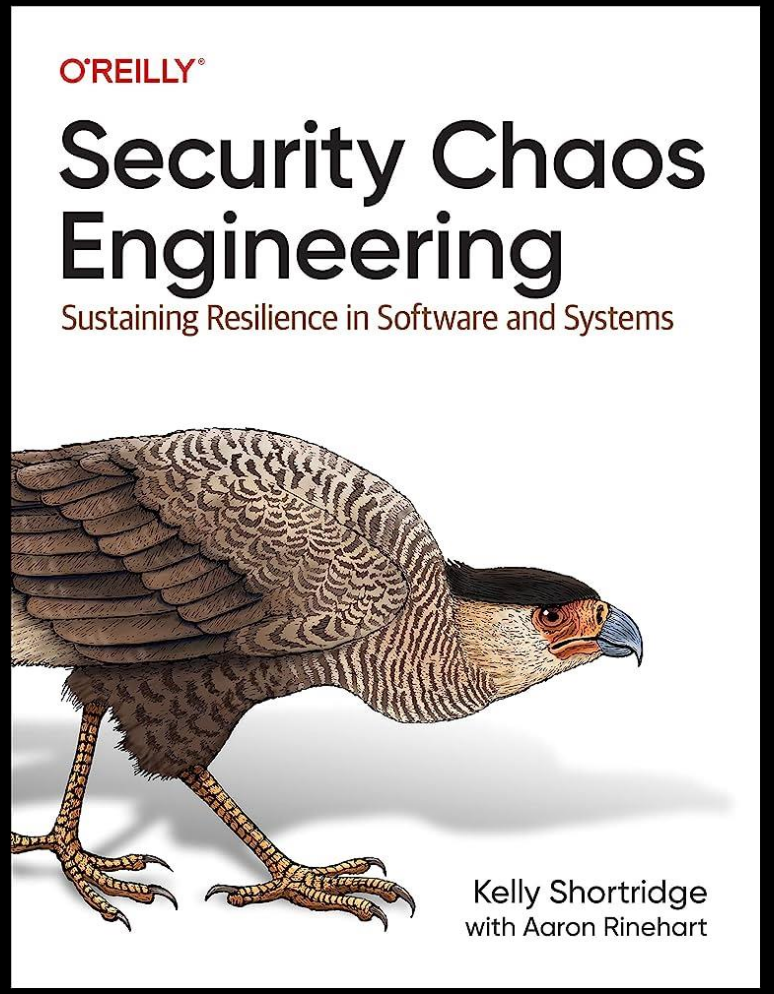

And that, comrades, is the resilience revolution.

Order the book today:

Amazon

Bookshop

& other major retailers



SHORTRIDGE@HACHYDERM.IO | @SWAGITDA_

/in/kellyshortridge

@swagitda_

shortridge@hachyderm.io

@shortridge.bsky.social

chat@shortridge.io