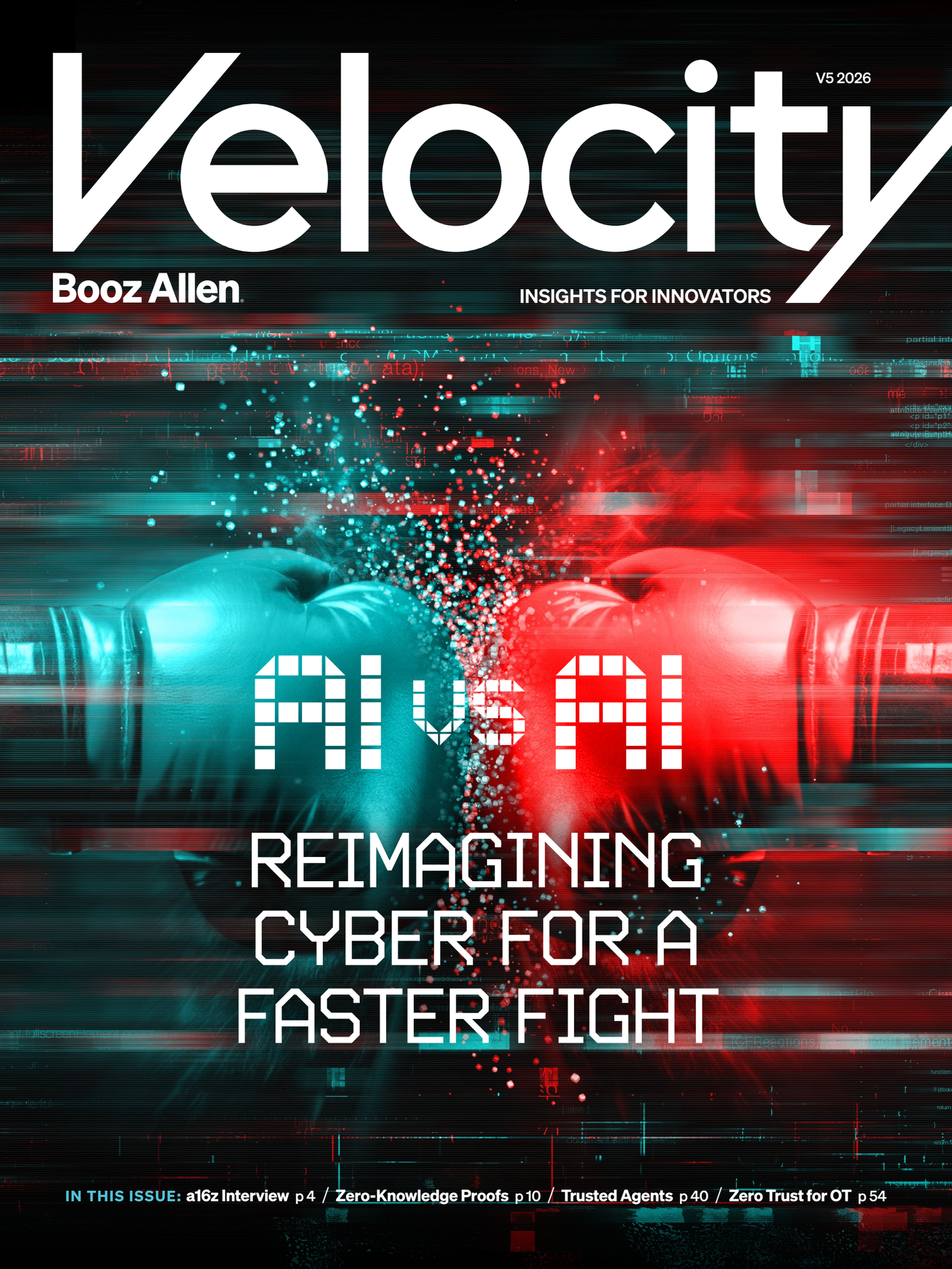
Velocity

V5 2026

Booz Allen

INSIGHTS FOR INNOVATORS



AI vs AI

REIMAGINING CYBER FOR A FASTER FIGHT

Resilience Tops Perfection:

DESIGNING FOR FAILURE WINS

Decades in IT have taught me: Failure is a normal operating condition

By Bill Vass

For decades, technology leaders have chased uptime like it was the ultimate prize—secure five nines reliability, flawless deployments, global availability. But here's the truth: FAILURE IS INEVITABLE.

After 48 years building and operating distributed systems—from the Pentagon to Sun Microsystems, startups, AWS, and now Booz Allen—I've learned something you can count on: Systems will fail. Maybe not today or tomorrow. But eventually, they will.

Networks fail. Hardware breaks. Air-conditioning units overheat. Power drops. Software behaves in ways no one predicted. Sometimes it's a bug, sometimes a storm, sometimes an adversary. The specifics don't matter—but your response does.

If a system isn't secure and available, none of its wonderful features matter. Resilience doesn't come from preventing failure. It comes from surviving it well.

That means limiting the blast radius using segmentation and thoughtful architecture—and recovering faster every time systems break, not pretending they never will. Learning that reshaped how I think about architecture, operations, and leadership.

I've earned a few scars along the way. I've seen what breaks systems—and what keeps them going.

LESSON 1:

Assume Everything Fails

It was a hard lesson, but now I know: All hardware fails.

Once you accept that failure is normal, your focus shifts. Instead of wondering if something will break, you try to manage how much breaks when it does. Software has to be written with the expectation that the underlying infrastructure will eventually let you down.

That was easier to manage earlier in my career, when systems were centralized and had fewer failure modes. Today's reality is different. Cloud, edge, AI pipelines, and distributed data stores multiply both scale and complexity. Every dependency introduces another opportunity for things to go wrong—and the more you scale, the more exceptional conditions you encounter.

That's why resilient systems don't rely on "strong" hardware or perfect networks. They assume the opposite. Stateless, load-balanced services are more forgiving by design. If one node fails, traffic shifts. If one system degrades, the rest keep running. The system isn't reliant on any single component to stay up.

In practice, this changes how you deploy. Physical infrastructure gets distributed wide enough to survive natural disasters, so a single event doesn't take everything down. After you finish alpha, beta, and gamma testing, you don't push code everywhere at once and hope for the best. You start with one box and give it load. You let it bake. If it fails, you roll back and fix it. When you're confident, you expand to one availability zone. Then another. Then you repeat, region by region—always with the option to fail and go back to the prior stage.

Building these steps into deployment helps you find failure early and limit the impact. High-reliability organizations like AWS don't depend on luck. They're methodical. They take small, deliberate steps—every single time.

LESSON 2:

Limit the Blast Radius

The most resilient systems I've seen are intentionally segmented. They isolate services, separate workloads, and assume trust must be continuously earned, not implied.

Practices like multi-availability zone and multi-region deployment aren't about redundancy for its own sake. They're about containment. Bulkheads keep one failure from sinking the entire ship. Circuit breakers stop bad dependencies from dragging healthy services down with them. Physical separation—10 kilometers or more between facilities—reduces the risk that a single storm, power event, or fiber cut takes everything offline at once.

Zero-trust principles reinforce the same idea from a security perspective: Never assume a component is healthy just because it exists inside the perimeter.

And then there's defense in depth. Real resilience takes more than a single layer of protection. It stacks multiple, independent, and heterogeneous layers—network paths, firewalls, encryption mechanisms—so a flaw in one doesn't compromise them all.

This matters even more in national security and critical infrastructure environments, where failure is operationally disastrous. Systems have to degrade gracefully, recover locally, and keep working long enough for humans to intervene.

LESSON 3:

You Can't Fix What You Can't See

Within minutes of an outage, I can tell whether a team will recover quickly or spiral. The difference comes from visibility, experience, and discipline. The best teams measure, rehearse, and learn every time something breaks.

Teams that recover faster have deeply instrumented systems. They know what "normal" looks like, so anomalies stand out immediately. Operations calls are focused, not chaotic, because the data tells a clear story about what's failing and where.

The best teams don't just restore service and move on. They investigate what failed, why it failed, and how to make sure it doesn't happen again. In most environments, a small number of root causes drive most outages. Resilient teams identify those causes, automate them away, and keep working down their Pareto charts until issues become less frequent and less severe.

That discipline doesn't just live in incident calls—it shapes how we build systems in the first place. Booz Allen-developed platforms like Recreation.gov are distributed across regions, instrumented from the start, observable under stress, and designed to learn every time something goes wrong.

Now, AI and automation are supercharging both the process and the payoff. AI-assisted operations can accelerate log analysis, find patterns humans might miss, and speed diagnosis when seconds matter. But AI only helps if systems are observable to begin with—it needs signals to work from. As instrumentation improves, so does AI's effectiveness. Over time, with the right inputs, agentic systems could move toward self-healing—detecting root issues and automatically initiating corrective action.

LESSON 4:

Define What Matters Before It Breaks

Resilience also requires realism about what matters most. Not every system deserves the same level of protection, and pretending otherwise wastes time and money. Before an outage happens, leaders need to classify their applications, defining which are mission critical, which are business critical, and which can tolerate some downtime. Those decisions shouldn't be made mid-crisis.

Every level of resilience carries tradeoffs in cost, performance, and complexity. Live-live replication across regions is expensive. Hot-cold deployments cost less but accept some downtime. Backup-and-restore models are even cheaper, but recovery may take hours. Decide ahead of time what risks you're willing to take. Resilience should be intentional, not reactive.

CONCLUSION: BUILD FOR REALITY, NOT PERFECTION

These lessons matter even more now.

Today's systems don't live in a single data center or cloud. They span hyperscale environments, on-prem networks, edge nodes, and devices operating far from reliable connectivity. AI and autonomy raise the stakes even further.

In national security and defense missions, systems have to keep operating even when the network is degraded, jammed, or gone entirely. Resilience can't depend on constant connectivity, perfect data, or centralized control.

It must be designed from the start—and strengthened iteratively over time.

And it often begins at a much smaller scale than leaders expect.

START WITH ONE SYSTEM THAT MATTERS.

MAP ITS DEPENDENCIES HONESTLY, NOT OPTIMISTICALLY.

INSTRUMENT IT SO YOU CAN SEE WHEN IT'S UNDER STRESS.

PRACTICE FAILURE—ON PURPOSE—BEFORE IT HAPPENS.

LEARN FROM EVERY INCIDENT AND BAKE THOSE LESSONS BACK INTO THE DESIGN.

REPEAT.

Resilience isn't a one-time investment or a line item in a budget. Organizations that thrive aren't chasing perfect uptime—they're prepared for imperfection.

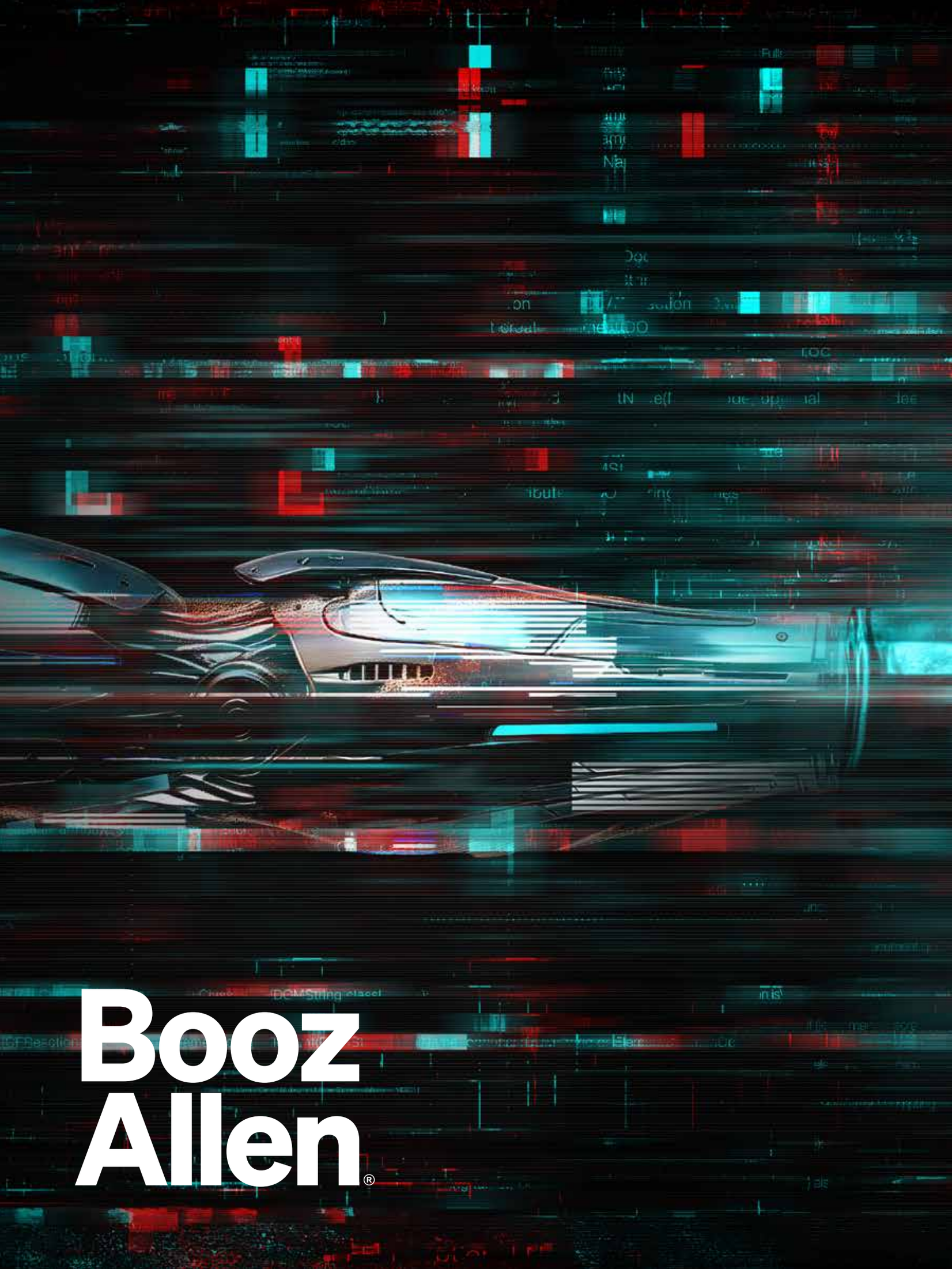
Bill Vass is the chief technology officer at Booz Allen—and an award-winning, visionary technology leader. He's built and operated distributed systems at massive scale for robotics startups, Fortune 100 companies, and the federal government.

SPEED READ

Because modern resilience assumes systems will fail, IT leaders need to focus on containing impact and recovering quickly—not chasing perfect uptime.

Good architecture limits the blast radius, while segmentation, zero trust, and multi-region design strengthen both operational resilience and security.

Teams that measure, rehearse, and learn recover faster—leveraging visibility, disciplined postmortems, and automation to turn failure into continuous improvement.



Booz Allen®