CERT PODCAST SERIES: SECURITY FOR BUSINESS LEADERS: SHOW NOTES

DevOps: Transform Development and Operations for Fast, Secure Deployments

Key Message: "Release early, release often" to significantly improve software performance, stability, and security using a DevOps approach.

Executive Summary

"Today, most organizations rely on IT more than they know. Almost every commitment a company makes to the outside world, whether it's related to projects, operations, compliance or financial reporting, requires IT. Most critical business functions are entirely automated within IT, and 95% of all capital projects depend on IT to get these done. Today, nearly every business decision will result in at least one IT change. And yet, the IT organization is often mired in urgent and unplanned work, starving their ability to execute. Spending countless hours dealing with emergency outages and failures and laboring over needless compliance projects, IT all too often gets stuck with damage control instead of deploying features and changes, and helping the business win" [1].

In this podcast, Gene Kim, founder and former CTO of Tripwire, discusses his experiences in observing and describing the unique behaviors of high-performing IT and security organizations. He discusses the benefits resulting from the DevOps (development and operations) method, used by organizations such as Netflix, LinkedIn, Google, Amazon, Twitter, and Etsy for fast, continuous, and more secure software deployments.

Gene and his two co-authors, Kevin Behr and George Spafford, illustrate these ideas in story form in their recent book titled *The Phoenix Project: A Novel About IT, DevOps, and Helping Your Business Win*.

PART 1: DOWNWARD SPIRAL, ESCALATING TECHNICAL DEBT, SECURITY AS COMPLIANCE, BURNOUT

Every Company Is an IT Company

Most organizations say, "We're not Google; we're not Etsy, so IT doesn't have to be a core competency." However, 95% of all capital projects have a technology component and 50% of all capital spending is technology related. Thus, technology is both the nervous system of a company and the majority of the muscle mass.

IT regularly appears to be in the way of an organization achieving its goals and aspirations. In addition, IT impedes project work to deliver new capabilities to the market as it attempts to balance this with delivering reliable, stable, secure IT service for existing capabilities. Most organizations think it is impossible to have fast feature flow to the market while preserving stability, reliability, and security.

Downward Spiral

The mission of IT is to provide reliable, stable, and secure service to customers. However, the problem is that so many of the assets it relies upon to do this are fragile, i.e., every time you try to patch or change one of these assets, something blows up – often catastrophically. This causes a huge amount of unplanned work.

Fragile artifacts typically support the most critical business operations and initiatives. As a result, organizations are often unable to meet their market commitments (features promised, earnings per share, etc.)

Organizations compensate for missing their commitments by making even bigger, more audacious promises. The development organization is then faced with urgent, date-driven projects, having to cut corners to meet due dates. This

results in:

- focusing on features instead of reliability, stability, and security
- using up all of the time in the schedule, leaving none for quality assurance, capacity and deployment testing, information security, and operations
- more fragile infrastructure in production

Escalating Technical Debt

As this downward spiral occurs, organizations accrue increasing technical debt (work that needs to be done before a deploy can occur), which means that deployments start taking longer (say, from 5 minutes to 1 hour to 1 day to 6 weeks).

With this situation, most of IT's time is spent firefighting and on unplanned work. This prevents the organization from doing more planned work to eradicate technical debt, which would reduce the surface area for risks caused by security vulnerabilities.

Once this happens, organizations become less able to make good on their promises and to survive.

Security as Compliance

Information security is often pushed to late in the life cycle due to its reputation of slowing things down, blowing up project due dates, and costing too much. Thus, development and operations groups tend to bypass security (if they consider this at all).

The result is that security is relegated to the role of compliance – get all the boxes checked from regulators, auditors, and certification and accreditation authorities. In this role, security is *not* making the organization more secure and is *not* helping it to meet its goals.

Burnout

Josh Corman did a study of burnout in the information security industry. He found that the levels of burnout in this profession are higher than those for first responders, physicians in emergency rooms, and those doing multiple tours of duty in the military.

The three signs of burnout are:

- fatigue
- cynicism
- illusions of self-efficacy (thinking that we're better than we are)

This is what happens to security teams by virtue of having to live with the consequences of decisions made upstream, including compounding technical debt.

This is not good for individuals, the profession, and the organization.

PART 2: IMPROVED CYCLE TIME, EARLY INVOLVEMENT OF SECURITY, TEN DEPLOYS PER DAY

The Promise of DevOps

DevOps includes a pattern of behaviors that can support a feature flow of hundreds and thousands of changes per day while achieving world-class security and reliability. Modern internet companies such as Google, LinkedIn, Twitter, and Netflix use DevOps practices.

DevOps provides an opportunity for security to become integrated into the daily work of development and operations. The cultural norms that are emerging from the DevOps approach include not only

- improved feature delivery rates
- reduced cycle time

but also

- quality
- scalability
- deployability
- manageability
- security
- visibility
- reduction of defects and vulnerabilities: time from vulnerability detection to vulnerability fix generate to vulnerability fix deployed becomes very short

John Allspaw, head of operations at Flickr, and Paul Hammond, director of engineering at Yahoo, kicked off the DevOps movement in 2009. Both organizations routinely did 10 deploys a day, every day.

Deploy Rates

At the 2011 Velocity Conference, Jon Jenkins at Amazon stated that they are now doing one deploy every 11 seconds. A deploy is any change that affects source code that is currently in production. It can be as small as a content change on a website or a new feature test and as large as thousands of new environments being deployed behind a load balancer.

By contrast, most organizations have one deployment and release every year – a difference of four orders of magnitude.

Distinguishing Deployments from Releases

Deploys occur continuously; releases typically occur at the end of a project. Waiting until the end to do integration and full feature testing combines deployment risk (making one big batch of changes to the production environment) with release risk. And this is being done when it matters most – on the day new features are promised to the market.

Deployments and releases need to be treated separately. When this is done, development and operations can be held responsible for deployment risk. Marketing is then responsible for release risk (did the new features achieve the desired result?).

With this approach, when a fix to a problem is identified, it can get into production in hours or days instead of having to wait until the next "release."

PART 3: THREE WAYS - ACCELERATE LEAD TIME, SHORTEN/AMPLIFY FEEDBACK, PRACTICE FAILURE FOR RESILIENCE

The First Way: The Way of Flow

Gene:'s work as reflected in *The Phoenix Project* is heavily influenced by the work of Eliyahu Goldratt, describe in his book, *The Goal*. The three ways are a set of principles from which DevOps patterns are derived.

In the first way, flow is from left to right (think of a timeline), from development into operations. The goal is to accelerate flow between the identification of customer needs to a business outcome, most often achieved by these 2 organizations.

The most important metric for accelerated flow and performance is lead time – from raw materials to finished goods and from code committed to code successfully running in production. For flow to be effective:

- work should flow quickly and smoothly in one direction
- it should not ping pong back and forth between dev and ops
- it should not be delayed by security

In high-performing organizations, lead time is measured in minutes or hours. For others, the mean in somewhere between months and quarters. Not only are high-performers deploying code three times more quickly; they are getting far better outcomes such as

- twice the change success rate
- mean time to repair being 12 times shorter
- greater product stability and agility

To achieve the first way, operational environments need to be available, on demand, to development and security, as an automated process.

The Second Way: Feedback

In the second way, flow is from left to right, from operations into development. The purpose is to get lessons learned in operations (such as failure modes) into development to create better quality at the source.

The objective is to shorten and amplify feedback loops and to create capabilities and knowledge where it is needed. One manufacturing example is the use of the Andon cord at Toyota. Every employee is empowered to pull a cord that stops the production line if there are deviations from expected behavior such as a defective part.

Another way to shorten feedback is to wake up developers at 2 a.m. when software they have developed breaks in production. Shared pain leads to shared goals. It is important that development teams live with the consequences of their decisions.

This approach also gets developers closer to their customers and allows them to experience the entire value stream.

The Third Way: Culture

The important principle here is constant experimentation (be allowed and expected to experiment) and learn from failure. Repetition is a prerequisite to mastery. An emerging school of thought is that it is better to practice 15 minutes a day every day than to practice once a week for 3 hours.

The annual security penetration test is an example of how *not* to do this. A much more effective approach is that used by Netflix, called Chaos Monkey, which allowed Netflix to be one of the few organizations that survived the large-scale Amazon EC2 failures in 2011.

The purpose of Chaos Monkey is to fail often to master the actions required to respond to and recover from failures. At Netflix, Chaos Monkey runs on every production server in the cloud and randomly kills processes and entire compute instances – in essence, it injects large-scale faults in production.

This approach creates more resilient code, infrastructure, and organizations. Failure becomes a part of everyday operations, which, as it turns out, is the norm even when you don’t plan for it. Resilience cannot be achieved without cost and adversity.

Resources

[1] IT Revolution Manifesto

- The Phoenix Project: A Novel About IT, DevOps, and Helping Your Business Win
- Resource list
- Twitter infosec talk
- Book excerpt (50%)

CERT Podcast: Connecting the Dots Between IT Operations and Security, May 2008.

CERT Podcast: Change Management: The Security 'X' Factor, November 2006.