

THE DEVOPS ROADMAP FOR SECURITY

*RUGGED PRINCIPLES, PRACTICES, AND
TOOLING FOR BRINGING THE SECURITY
TRIBE INTO DEVOPS.*

The DevOps Roadmap for Security
Rugged Principles, Practices, and Tooling for Bringing the Security Tribe Into DevOps
by James Wickett and Signal Sciences

Copyright © 2016 Signal Sciences. All rights reserved.
Published by Signal Sciences, 122 Mildred Ave, Venice, California 90291.

Version Information

This is the 1st Edition of the The DevOps Roadmap for Security.
This document was last updated on July 29th, 2016.

THE DEVOPS ROADMAP FOR SECURITY

Rugged Principles, Practices, and
Tooling for Bringing the Security Tribe Into DevOps

JAMES WICKETT
SIGNAL SCIENCES

Table Of Contents

Recognition	5
Introduction	6
Why DevOps Matters to Security	7
DevOps Adoption	7
Security's Problem	8
Unifying the Tribes	8
Infrastructure as Code	9
Version Controlled Artifacts	10
Configuration Management	10
Testing	10
Cloud and Distributed Computing	11
Software Supply Chain	11
Summary of Infrastructure as Code	12
Engineering Culture	13
Business Impact of DevOps Culture	13
What culture means to security	14
Pragmatic Technical Changes	15
Lean Security and Eliminating Waste	15
Democratization of Security Data	16
Delivery Cadence	18
Why is Delivery Cadence so Important?.....	18
Three common practices with implications to Security	19
1. Smaller Changes are Easier to Rationalize	19
2. All the Testing is Automated	19
3. Assurance and Confidence in changes	20
Feedback Loops	21
A Defensive Thinking Approach	21
Application Security Feedback	22
Usage Feedback	22
Summary	24

Recognition

"Security is joining forces with DevOps and this paper shows you how to get started with common principles and practices to effectively integrate security in your DevOps transition, written by some of the best in the game."

Gene Kim, co-author of "The Phoenix Project" and the upcoming "DevOps Handbook"

Introduction

DevOps is a culture, movement, and practice that enables collaboration between development and operations teams throughout the entire software delivery lifecycle, from design and development to production support. It breaks down entrenched silos, allowing organizations to transition from functional area delivery to a more holistic approach. This results in robust processes, exponential improvements in deployment times, and ultimately, superior results to a company's bottom line. Since DevOps was first coined in 2009¹, it has been a massive movement among engineering focused organizations. In the 2015 State of DevOps report, it was found that high-performers were able to deliver *thirty times* more frequently--rapidly decreasing the time from concept to cash.^{2 3}

DevOps is transformational in four key areas:

- It treats all systems and infrastructure as code
- It shifts engineering culture towards total delivery and user experience.
- It favors a faster delivery cadence and a reduction in changes per delivery.
- It creates feedback loops in the runtime environment to inform development and operations.

In each of these four areas there is a common body of principles, practices, and tooling that are rapidly evolving. *The DevOps Roadmap for Security* will help you navigate these areas and suggest ways for security teams to get more involved with DevOps.

¹ What Is DevOps? (2010). <https://theagileadmin.com/what-is-devops/>

² 2015 State of DevOps Report. <https://puppet.com/resources/white-paper/2015-state-devops-report>

³ Poppendieck, M., & Poppendieck, T. D. (2007). Implementing lean software development: From concept to cash. Upper Saddle River, NJ: Addison-Wesley. DevOps borrows heavily from Lean and moving from Concept to Cash was popularized in "Lean Software Development: From Concept to Cash".

Why DevOps Matters To Security

DevOps Adoption

In early 2015, Gartner, a leading market research firm, stated that: “By 2016, DevOps will evolve from a niche strategy employed by large cloud providers to a mainstream strategy employed by 25 percent of Global 2000 organizations.”⁴ It’s safe to assume that if you aren’t considering DevOps now, the market may soon decide for you. According to a survey and published report by CA⁵, there are five key benefits to DevOps adoption:

- **Provides new software or services that would otherwise not be available.** This arms organizations with a new playbook for cloud delivery, microservices, and software as a service offerings.
- **Reduces time spent fixing and maintaining applications.** DevOps practices have proven to require less break-fix work and decreases the mean time to recover (MTTR) from outages.
- **Improves cross departmental collaboration.** DevOps is born from collaboration across functional silos, and organizations that adopt it are witnessing the benefits first hand.
- **Increases revenue.** In his book, *Leading the Transformation: Applying Agile and DevOps Principles at Scale*⁶, Gary Gruver directly tied DevOps transformations to bottom-line impact in HP’s printer business by reducing costs by \$45M and freeing up 35% capacity for new innovation. Bringing both at the same time made a significant impact to the business.
- **Improves quality and performance of deployed applications.** One other tangible outcome of DevOps is a reduced failure rate. The 2016 State of DevOps report showed that high-performers had a 3X lower rate of failure.⁷

⁴ Gartner Says By 2016, DevOps Will Evolve From a Niche to a Mainstream Strategy Employed by 25 Percent of Global 2000 Organizations. <http://www.gartner.com/newsroom/id/2999017>

⁵ CA Research Report: DevOps: The Worst-Kept Secret to Winning in the Application Economy. October 2016.

⁶ Leading the Transformation: Applying Agile and DevOps Principles at Scale Paperback – August 1, 2015. <https://www.amazon.com/Leading-Transformation-Applying-DevOps-Principles/dp/1942788010>

⁷ Puppet. (2016). 2016 state of DevOps report. <https://puppet.com/resources/white-paper/2016-state-of-devops-report>

The benefits of DevOps are clear for organizations of all sizes, and the adoption rate suggests that even more evidence will be forthcoming over the next few years.

Security's Problem

While DevOps adoption is on the rise, the same can't be said for security success. In his latest book, *"Thinking Security"*, published at the end of 2015, Steven Bellovin writes, *"Companies are spending a great deal on security, but we still read of massive computer-related attacks. Clearly something is wrong."* This assessment is validated by the daily media deluge of security breaches and failings of major companies, some of which specialize in security. Security is not only failing to protect, it's also hindering the organization's productivity. Bellovin goes on to write, *"The root of the problem is two-fold: we're protecting (and spending money on protecting) the wrong things, and we're hurting productivity in the process."*⁸

It doesn't have to be this way. In fact, there are many organizations that are integrating security with DevOps. Contrary to popular belief, Security and DevOps have a lot more in common than people think. Both tribes benefit immensely from each other, sharing in the outcomes of their collaboration.

"Security is not only failing to protect, it's also hindering the organization's productivity."

-- Steven Bellovin

Unifying the Tribes

We've been using the term "tribes" throughout this report. To add clarity, tribes are simply groups of people found in organizations that rally together around common concerns. DevOps is concerned with uniting two particular tribes: **Development** and **Operations**. These tribes have seemingly opposing concerns: developers value features while operations value stability. However, these contradictions are largely mitigated by DevOps. A strong argument could be made that the values of the Security Tribe--defensibility--could just as easily be brought into the fold, forming a triumvirate under the DevOps umbrella.

The Security Tribe's way forward is to find ways to unify with DevOps in its four key areas: (1) Infrastructure as Code, (2) Engineering Culture, (3) Delivery Cadence, and (4) Feedback Loops. Each of these areas will be covered in the following sections of this report.

⁸ Bellovin, S. M. (2015). *Thinking security: Stopping next year's hackers*. United States: Addison-Wesley Educational Publishers.

Infrastructure As Code

The first area to explore in DevOps is Infrastructure as Code. This is the complete codification of networking and routing to the system configuration. Everything that is needed to create, run, change, and destroy infrastructure is expressed in code

The broader goals of Infrastructure as Code include:

- **Version controlled artifacts that describe the system and all its components.** This keeps configuration out of wikis and documents and in a versionable, referenceable state.
- **Configuration management of the system in running state.** Configuration and runtime state tracking replaces CMDB.
- **Testing as a first order priority with Test-Driven Development (TDD) and integration testing as common practices.** Tests are written for infrastructure code as well as application code while under development. Writing tests while creating your infrastructure both asserts desired state as well as provides a test suite for CI/CD efforts.
- **Facilitating distributed computing and scaling.** Without treating infrastructure as code, scaling is difficult and distributed computing (cloud) becomes almost untenable. Seeing distributed computing and scaling as desired outcomes guides the development practices.
- **Understanding your Software Supply Chain.** Software is not merely the hundreds or thousands of lines of code that are written by developers, in reality it is composed of much more from dependencies to the OS to the virtualization framework. Infrastructure as Code encourages software supply chain management by introducing specificity and an auditable log for the actual runtime of the system.

Some practical artifacts of adopting Infrastructure as Code include Dockerfiles, Terraform Plans, or Chef cookbooks. The use of such artifacts will change based on the underlying infrastructure shifts from bare iron to virtual machines, to public cloud services, and now to containers and serverless patterns. What will stay the same are the broad principles outlined above. Let's cover each principle

in greater detail:

Version Controlled Artifacts

Having version controlled artifacts is one of the first steps to doing Infrastructure as Code. These artifacts bring in the core functions of auditability and change control to the operational process. Version Controlled Artifacts includes version controlling all of the configuration management code, but also creation scripts and image packaging code. Often there are other pieces of the infrastructure that need to be added that can't operate as readable artifacts. These are components like ssl wildcard certs, license files, or passwords. These often will be still version controlled but only in encrypted binary form.

In Gene Kim's book on the topic of change control, Visible Ops Security, Gene demonstrates that there is a direct cause-effect relationship between the ability to detect change in security components and the success of security initiatives. With operations moving into version control (just like development), the security team now has a foothold and view into the entire system. This encourages change control with alerting of changes to critical components and auditability that was never available before.

Configuration Management

Configuration management expresses the configuration of the running system in code. Convergence and idempotency are the two core concepts behind configuration management.

- **Convergence:** assures that the infrastructure will reach its desired state through the configuration management system.
- **Idempotency:** guarantees that a command can be run over and over with the same results. Because configuration management has both of these attributes, there can be better reasoning around the system.

From a security team's perspective, there are two key benefits to configuration management. First, configuration becomes accessible in an easy-to-read, federated format. This simplifies auditing and allows security to have insight into how the systems are built -- complete with logs. The second benefit is compliance and adherence to policy. Enforcement of policy is a fundamental requirement for successful security. Configuration management allows security teams to get much closer to reaching this goal in an automated fashion.

Most configuration management systems have built-in functionality to run in validation mode rather than to attempt convergence. Running configuration management in validation mode allows verification of the system on a daily (or more frequent) basis, ensuring it hasn't drifted out of compliance. The popular configuration management system, Chef, provides this exact benefit via Inspec. Inspec is an open-source testing framework for specifying compliance and policy requirements.⁹ Inspec provides a huge advantage by creating daily reports of runtime drift out of compliance or even more importantly, identifying new exposure areas.

Testing

There are two main types of testing relevant to infrastructure as code: Test Driven Development

⁹ Chef Software 2016. <https://www.chef.io/inspec/>

and Integration Testing. Test Driven Development means that the developer writes tests alongside the development of the application or, in the case of Infrastructure as Code, the infrastructure. The benefits of Test Driven Development are numerous, but one of the key improvements is the creation of a functional test suite that can be used with continuous integration / continuous deployment efforts.

The second type of testing is Integration Testing. This is an outside-in approach of asserting that the infrastructure and system meet the requirements set forth at the time of design. Tools like Serverspec, KitchenCI, or Robot Framework are often used to do this layer of testing. It's tempting to think that integration testing is done at a later stage in the software development lifecycle. However, there is a growing trend of shifting this testing "left," or earlier in the development and delivery pipeline.¹⁰

Security testing does not have to be much different to run earlier in the lifecycle. In fact, the industry is continuing to move security testing further left in the pipeline using tooling like Gauntlt or Mittn. With these security-centric testing frameworks you are able to specify the security standards you expect all software to meet. For example, "our website should not fail a scan for XSS" or "when not logged in you should not be able access certain resources." Once the definition of the requirement is set, you can implement automated integration testing and test driven development to ensure success.

One company taking this approach is ThreatStack and they actively look for changes with security implications both at the host and the cloud provider configuration layer.

Cloud and Distributed Computing

In the modern world, we often find ourselves running our systems on third-party providers like AWS, Azure, or Rackspace. Running in these cloud providers changes how we think about security incidents and lateral movement. Cloud computing changes our threat landscape. Attackers are less likely to gain a foothold by pivoting across your systems through network segments, but instead will attack your cloud provider's configuration and seek to open holes in that environment.

One key weakness that most enterprise security departments have is that they are not prepared to deal with this new cloud landscape. In fact, industry experts have dubbed it a "black hole" due to the disconnect that they feel.¹¹ To deal with this, cloud providers like Amazon Web Services (AWS) provide a complete audit log. For AWS it's called CloudTrail, which logs all changes to every single configuration in your cloud architecture. Meanwhile, auditing monitors all system commands run on the hosts. Combining these two vectors of logging and auditing provides a clearer picture to changes happening throughout the environment.

Software Supply Chain

Software is not merely the hundreds or thousands of lines of code that are written by developers. In reality, it is composed of much more, from individual dependencies, to the operating system, to the virtualization framework. Unfortunately the software we build inherits vulnerabilities from all the code we didn't write.

¹⁰ Shift left testing (2016). https://en.wikipedia.org/wiki/Shift_left_testing

¹¹ Ashford, W. (2016, July 13). DevOps a black hole for security. <http://www.computerweekly.com/news/450300208/DevOps-a-black-hole-for-security>

Infrastructure as Code encourages software supply chain management by introducing specificity and an auditable log for the actual runtime of the system. Knowing what code we are shipping through a Software Bill Of Materials (Software BOM) is an important engineering and security task. Being able to know what is in your current runtime down to the specific version is critical. This includes all included code libraries as well as sub libraries that your inclusions use. Software supply chain is a difficult problem to solve due to the redundant nature of code reuse.

Summary of Infrastructure As Code

Infrastructure as Code has a lot of implications to security and is often one of the most pragmatic starting points for security teams that want to become involved with DevOps. However in the case of many organizations, the problems they face aren't technical so much as they are cultural in nature

A solution like Sonatype can be immensely helpful to understand your Software BOM

Engineering Culture

Culture is the foundation of any business function, and that is especially true for DevOps. In fact, many of DevOps' early adopters define it first and foremost as a cultural movement followed by operational and technology requirements. The adherence to a culture-first approach to DevOps was an outcropping of the organizational divide between development and operations. The cultural divide was often apparent just by examining an organizational chart. With staffing ratios of ten developers to one operations staff, coupled with different chain-of-command paths, it was easy to pinpoint the source of the problem. There were also competing priorities, between stability (operations) and features development (development). This tension created silos based on functional roles in many organizations.

"You can't directly change culture, but you can change behavior, and behavior becomes culture."

—Lloyd Taylor VP Infrastructure, Ngmoco

Before DevOps had a name, it was originally referred to as Agile Infrastructure. Agile was successful at transforming development practices and behaviors. It seems obvious now that if the same agile principles were applied to operations, the cultural divide could be resolved. Due to the close relationship between Agile and operations, behaviors did start to change. New practices arose like Scrums, Kanban boards, standups and planning poker sessions. These collaboration practices were evidence of the behaviors that would eventually influence cultural change.

Business Impact of DevOps Culture

As DevOps amasses a larger following, the industry has begun to quantify the cultural impact that

DevOps makes. The *2016 State of DevOps Report* released a significant finding on culture: “Employees in high-performing teams were 2.2 times more likely to recommend their organization as a great place to work.” This statistics alone demonstrates the potential for culture change using DevOps concepts, however even more interesting stats help solidify the fact that DevOps is a business enabling ideal.

*Employees in high-performing teams were **2.2 times more likely** to recommend their organization as a great place to work.*

— 2016 State of DevOps Report

Employees working in high performing teams are twice as likely to recruit friends to work with them. Improved recruiting means that the employees are not just happy, but proud to associate their personal brand with the company. This is a direct reflection of the culture of the organization and the fact that the team is effectively accomplishing their goals—it comes down to having personal pride in their work, something that everyone strives for. It’s one thing to work somewhere, but it’s a completely different thing to ask your friends to join you.

What Culture Means To Security

If culture is the foundation of DevOps, and solving the cultural divide is important, then security should take notice. It’s easy for security to identify with the problems between developers and operations—security faces the same cultural issues. An average staffing ratio of one hundred developers to one security engineer illustrates an even larger divide than that which exists between developers and operations. Alongside this inequitable distribution of labor, there is the very real challenge of differing priorities: speed and features vs. defense and compliance.

100:1

A normal staffing ratio of developers to security staff.

Source: Shannon Lietz, Director of DevSecOps at Intuit
Safely Removing the Last Roadblock to Continuous Delivery – DevOps Austin Presentation

As security makes the cultural transition to DevOps, security professionals must:

- **Recognize that if security blocks progress and speed, it will be ignored and marginalized.**
Building or fostering a culture of gating functions surrounding security is not a sustainable or forward thinking model. Security must get out of the way of progress in order to survive.
- **Collaborate across the organizational landscape and deputize security champions.**
Enterprise security can’t be solved by simply hiring additional resources. There simply isn’t enough security talent available to fill the current needs, let alone future growth. Instead, the most effective security organizations are discovering ways to deputize security

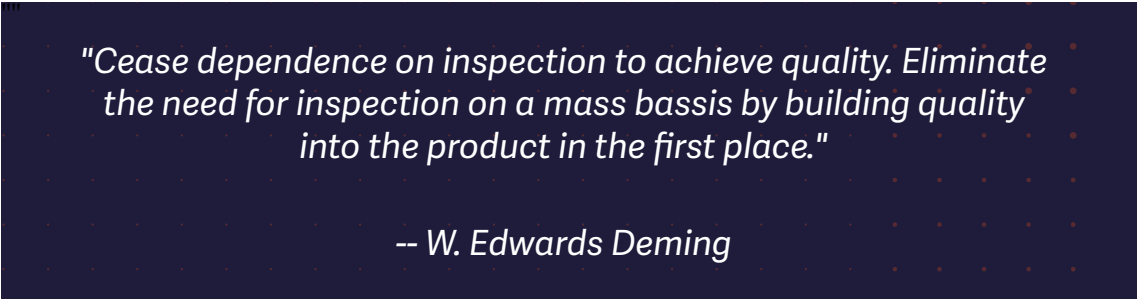
champions across their organization.¹²

Pragmatic Technical Changes

It's often said you don't fix cultural problems with technology. In general, this statement is true, however, there are some technical changes that can influence cultural behavior. For example, the *2016 State of DevOps Report* highlights one development pattern that actually led to higher performance and a better culture: the practice of trunk based development. This development pattern consists of having branches or forks in low quantities, with short lifetimes, and is an important aspect of continuous delivery. Lowering branch count was found to contribute to fostering a much higher performing team.

For security engineers there are two practices that impact culture: a Lean Security approach to eliminate waste and the democratization of security data.

Lean Security and Eliminating Waste



"Cease dependence on inspection to achieve quality. Eliminate the need for inspection on a mass basis by building quality into the product in the first place."

-- W. Edwards Deming

Currently, the entire software security industry is built on inspections at the end of development through processes like annual penetration testing or compliance assessments. This model runs counter to Lean Software Development practices.¹³ Using annual cycles and end-of-cycle inspection is harmful because it creates waste, delays learning, and slows down overall delivery.

One of the first considerations of a Lean Methodology is identifying waste and eliminating it from your production. At RSAC 2016, Ernest Mueller and James Wickett presented on Lean Security and how to identify waste with security practitioners in mind.¹⁴ Security professionals should focus their process improvement energy on lowering or otherwise improving:

- **Excess Inventory:** Caused by handing off a thousand page PDF of vulnerabilities to an already busy team. Excess inventory can be solved by prioritization and limiting the Work In Progress (WIP) queue. Focus on attainable goals that don't overwhelm your staff.
- **Overproduction:** Security controls stemming from Fear, Uncertainty, and Doubt (FUD)--the lingua franca of security--cause a misalignment with actual business need. Choose instead to align with actual need and eliminate ideas that can't be solved and only lead to confusion and FUD.

Extra Processing: Relying on compliance testing cycles as opposed to designing processes to eliminate problems from inception is a major issue. Security practitioners must decide to be

¹² More Silo Smashing Ideas, Bringing InfoSec and DevOps Together — Signal Sciences Labs, April 5, 2016, <https://labs.signalsciences.com/more-silo-smashing-ideas-bringing-infosec-and-devops-together-c338eb3b36ad>

¹³ Lean Software Development practices were created by Mary and Tom Poppendieck in their book *Lean Software Development: An Agile Toolkit* (2003).

¹⁴ Lean Security - RSA 2016 by Ernest Mueller. <http://www.slideshare.net/mxyzplk/lean-security-rsa-2016>

involved in the earlier stages of software creation rather than post-development testing.

- **Handoffs:** Handing problems to others to solve instead of collaborating and being a part of the solution limits collaboration and long term effectiveness. Solve problems, don't pass the buck.
- **Waiting:** Lag time between value steps such as waiting for approvals or analysis for security fixes impedes the goals of the business. Create self-service flows by automating security tooling, thus lowering the impact on development and operations.
- **Task Switching:** Rapid "break-fix" work, or hot patching should be avoided as much as possible. Security should adapt to use the current "work intake" processes that the development team prefers. Whenever security can operate within the confines of the current operational model they should strive to do so.
- **Inaccurate Defects:** Both false positives and false negatives are unimportant findings that often get reported, resulting in zero-value rework items and a waste of development and operational resources. Validate findings before reporting them to the team and make sure they are legitimate to streamline software security improvement.

Lean Software Development Practices:

1. ELIMINATE WASTE
2. AMPLIFY LEARNING
3. DECIDE AS LATE AS POSSIBLE
4. DELIVER AS FAST AS POSSIBLE
5. EMPOWER THE TEAM
6. BUILD INTEGRITY IN
7. SEE THE WHOLE

The process of finding waste and eliminating it in your system will increase productivity and boost culture. Shifting security engineering efforts earlier in development is not just about the removal of waste -- it has plenty of cultural benefits as well. Making gains both on your culture and your productivity will give your teams a one-two punch of security improvement.

Democratization of Security Data

The rise of DevOps has spurred other sub-movements, one being ChatOps. ChatOps is the practice of integrating your monitoring, logging, and other operational tasks into the team communication medium. Many organizations achieve ChatOps goals via IRC or other chat client systems like Slack or Atlassian HipChat. Almost every piece of tooling you use integrates in some way -- from deploys to monitoring to new customer signups -- with the leading technologies in the ChatOps space.

From a security vantage point, many teams have benefited from integrating their security tooling into their development and operational ChatOps efforts. Security and ChatOps integration takes the silo'ed knowledge of where attacks are happening and distributes it across the organization, opening up major lines of communication. As you might have guessed, security plus ChatOps has changed the way security is perceived at many enterprise organizations by quickly turning the

ChatOps concept into a security centered cultural win.

At Signal Sciences, we distribute security events to the entire team through methods that encourage collaboration. Many of our customers integrate with Slack and Atlassian HipChat or alerting products like VictorOps and PagerDuty

.....

Delivery Cadence

Continuous Integration and Continuous Delivery (often referred to collectively as CI/CD) is not a wholly new concept. However, the growth and adoption rate of CI/CD is on the rise. The focus on delivering software rapidly is influenced by the rapid growth of DevOps, and in many ways the delivery cadence metric of an organization is an indicator of how successful your organization has been at adopting DevOps. This doesn't mean that faster is always better. Success is better measured by reducing the Mean Time Between Delivery (MTBD) for your organization. Moving from monthly to weekly to daily delivery is a journey and it is a journey that's worth making.

*High-performing IT organizations experience **60X fewer failures** and recover from failure 168X faster than their lower-performing peers. They also deploy 30X more frequently with 200X shorter lead times.*

— 2015 State of DevOps Report

Why is Delivery Cadence so Important?

In the 90's and throughout the early 2000's, most of IT followed a waterfall model for delivering software. This means that software spends the majority of its time in architecture and design and only towards the end of development does it actually come together to function. The window for design and development could easily be 6-12 months, or longer, with the last month being the integration phase where it all gets connected and run together to be tested as a final unit. In many cases this would be the first time it would all come together to function as a whole.

The theory behind waterfall development is that if all the requirements were gathered first to

specify all the development tasks upfront, then at the end it would produce the correct result within budget and on time. Effectively, all changes are batched together into a release in the latter stages of the software engineering effort. Since this is quite an undertaking to deliver this batch of changes, it's untenable to do it that often. This causes releases to happen twice a year or even as slow as once every twelve months in many organizations. Since releases are so infrequent, it also encourages stuffing as many changes as you could fit into each individual release.

It was also believed that waterfall would result in less rework due to upfront specificity and increased stability and security since all changes are made in large batches. Today, the industry has realized this type of thinking is incorrect.

The *2016 State of DevOps Report* found that high performing organizations actually deploy 200 times more frequently than their peers resulting in faster recovery times, less rework and faster cycles from concept to cash.

Additionally, security gets better from faster delivery rather than worse. The report continues, "We found that high performers were spending 50 percent less time remediating security issues than low-performing organizations. In other words, because they were building security into their daily work, as opposed to retrofitting security at the end, they spent significantly less time addressing security issues."

The two key tools that need to be in place to influence delivery cadence:

- **CI/CD System.** Using a CI system like Jenkins or a service like TravisCI or CircleCI that runs tests and creates artifacts that can move on to the next stage in the delivery pipeline.
- **A deployment system with minimal gates that handles orchestration.** Seamless deployment across various groups and processes will ultimately save more time and lower any potential risks.

CI/CD systems are available as a service with two great options as the market leaders: TravisCI and CircleCI

Three common practices with implications to Security

There is an excellent book titled *Continuous Delivery* by Jez Humble and David Farley that is the comprehensive and definitive work on the subject. This book defines three specific practices that improve security:

1. Smaller Changes are Easier to Rationalize

One of the benefits of having a higher frequency of deploys is that you will also have smaller amount of changes going out each time. This makes each deploy simpler and easier to rationalize with regards to what is being changed. This has the side effect for security to isolate changes made to the more sensitive portions of the code base.

2. All the Testing is Automated

Continuous Delivery pipelines hinge on automated testing. Each commit, no matter how small, goes

through the same testing before getting released to a pre-production environment and ultimately to production. This is a good thing and security can take advantage of this playing field by adding in static and dynamic security tooling (SAST and DAST respectively) to the pipeline.

3. Assurance and Confidence in changes

One of the core tenets of Continuous Delivery is that artifacts are only built once and, as much as possible, are immutable. Continuous Delivery pipelines track the outcome of a build (the artifact) to a repository, to completing testing, to deployed in production. This increases confidence and assures the security team there is an audit chain for changes.

Feedback Loops

Feedback loops aren't a new idea—they are almost so inherent, so human, that it feels odd to specifically call them out at times. Of course there needs to be a feedback loop. Feedback loops are included in all human relationships and even with complex systems like automobiles or indoor heating and cooling systems. In military strategy there are OODA (Observe, Orient, Decide, Act) loops¹⁵ and in the best selling book *"The Lean Startup"*, the feedback loop is identified as the Build-Measure-Learn cycle.¹⁶ Yet somehow, software development struggles with defining its feedback loops. It is still common, though increasingly less so, to have production software where users report outages before the development, operations, and security staff are even aware of it. If security is to be successful in the new, shorter, DevOps cycles, feedback loops have to improve.

Once an organization has shifted thinking and processes to orient around a fast delivery cycle, the security team will need to quickly adapt. It becomes even more important to have insight into the rapidly-changing runtime environment. Security events become visible through feedback loops and instrumenting the runtime environment.

A Defensive Thinking Approach

At the risk of over-simplifying security concepts, defense requires knowing answers to two first-order questions:

- Am I currently being attacked?
- What vector of attack is being attempted?

This is further complicated by second-order factors such as analyzing the likelihood of success

¹⁵ Theory in practice — OODA, mapping, Antifragility. (2016, July 11). Retrieved July 26, 2016, from <https://medium.com/@aneel/theory-in-practice-ooda-mapping-antifragility-df7f03a36a9c>

¹⁶ Ries, E. The Lean Startup. <http://theleanstartup.com/principles>

and determining the potential cost of compromise. The security industry at large generally isn't equipped to address these questions due to a lack of first-order data, namely their limited insight into the frequency and types of attacks. Surprisingly, most organizations can't even approximate an answer to these questions.

In a DevOps context, there are three areas where security provides direct value to the enterprise, utilizing value available from integrations across the organization. Each of these areas can give insight into the first-order questions and through instrumentation, can shift to a defensive thinking approach. When providing security defense in a web context there are two key areas to evaluate: application security and usage feedback.

Application Security Feedback

It's hard to think about modern approaches to delivering services without thinking about delivering them over the web. With the rise of Microservices and the decoupled architecture patterns therein, you find an even higher dependence on web-based REST APIs. Today most systems are really collections of loosely coupled applications delivered over the web. This hasn't been an abrupt transition, but has been an ongoing shift over the last 20 years. But even with a long history of using the web, we have a dearth of mechanisms for detecting security problems in real-time.

Many organizations implemented Web Application Firewalls (WAFs) a decade ago, however rarely has anyone operationalized them. Most WAFs were put in place for compliance adherence, namely PCI, and were generally put in listening mode with no defensive posture. However, in the last ten years we have continued to see common web application security vectors get compromised and the Open Web Application Security Project (OWASP) continues to issue guidance on the same threat vectors. The problem hasn't been solved. We are clearly lacking feedback loops to improve on our application security stance in the face of changing underlying technology models.

There are two main feedback loops to implement in application security: divergent patterns and known attacks. Divergent patterns, or signals, are seen in traffic that perhaps attempts to access resources that don't exist or spikes in traffic from uncommon sources. Known attacks are common OWASP Top Ten items like XSS or Injection attacks. Feedback loops in both areas bring visibility to an otherwise neglected aspect of our systems.

Real time attack and event data are required for application security success. NGWAF and RASP technologies, when properly integrated with a DevOps tool chain can provide the feedback necessary to make informed actionable security decisions.

Usage Feedback

Customer usage is an often overlooked, yet very important, feedback loop for security and DevOps. Are you experiencing a higher volume of logins? What about password changes? Have you seen more accounts created in the last hour than normal? These are all subjective questions that are specific to the current business state. More than likely some of these metrics are already being tracked within the organization but they are not visible throughout. Enterprise security teams

should use these feedback loops to check for anomalous behaviors that are indicators of current or successful attack signals.

When combined with application security feedback, usage metrics become more powerful. Often these will give clues to how successful the attacks are. If there is a spike in XSS attacks, it is a more powerful metric when correlated with the number of password change requests happening in the application. Instrumenting the common flows for users in your system and tying them to application security feedback can bring tremendous value to all sides: development, operations, security and most importantly, the business.

Summary

The DevOps Roadmap for Security was written to help provide guidance to security practitioners that may be experiencing the transition to DevOps in their organization. Making changes to move forward and unite the DevOps and security tribes radically changes an organization's engineering culture. It changes the perception of infrastructure as well as shifts thinking around how fast a service or system should be delivered. Lastly, there is a new appreciation to feedback loops in the runtime systems with a focus on taking a defensive thinking approach.

