DevOpsSec: Security as Code

DevOps安全标准: 代码即安全

The approach that most financial organizations take to IT security

today is “scan, then fix.” They depend heavily on security reviews in

Waterfall project gates: reviewing specifications and architecture,

scanning code before it’s handed off to test, pen testing the system

before it goes live.

今天大多数金融机构对IT安全的做法是：“扫描，然后修复” 这个严重依赖于各项目之间的防火墙：

审视标准和架构，在提交到测试系统之前审视代码，在投入生产前测试系统。

But in DevOps there are no Waterfall gates where security audits or

penetration tests can be scheduled. There aren’t even any Agile security

sprints or hardening sprints. Security needs to be brought into

development and operations, and included in Continuous Delivery stages.

但是在DevOps ，这里没有安全审计或者渗透测试。DevOps甚至没有智能安全测试或者压力测试。

在持续交付阶段，在开发和操作的阶段，安全就需要植入其中。

Whether it’s called DevOpsSecor or “Rugged DevOps,”

or has a name at all, security in DevOps is based on a few key ideas and practices:

是否有DevOps安全人或者麻烦的DevOps，这类人也许连名字都没有。

DevOps在安全方面主要是基于一些关键的想法和实践。

1. Breaking down walls between development, operations, *and*

information security and compliance and risk management

teams, and bringing them all together to solve (and, more importantly, to prevent) security problems

1.推倒开发、运维、信息安全、合规、风控管理团队之间的那些墙，让这些团队协同去解决(更重要的是去预防)安全问题。

2. Shifting security controls and checks earlier into design and development

2.将安全控制和检查前移到设计和开发阶段

3. Automating security testing and security checks in Continuous Integration and Continuous Delivery, including security checks on dependencies

3. 自动化安全测试和安全检查包含在持续整合和持续交付，以及安全的依赖性检查中

4. Taking advantage of Infrastructure as Code and Continuous Delivery to standardize and secure the runtime environment

4. 利用基础架构即代码以及持续交付去标准化和安全化运行环境。

5. Leveraging the logging and workflow controls in Continuous Delivery to provide an audit trail of security checks for regulators

5.借助持续交付中的日志和流控制去为规则的制订者提供安全审计流

6. Wiring security into application operations monitoring and feedback loops.

6. 将安全交织在运维监控和反馈循环中

Let’s look at these practices in more detail, to understand what needs to be done.

现在让我们来看看这些实践的细节，然后明白什么是我们应该去做的。

Shift Security Left

To keep up with the pace of Continuous Delivery, security has to be “shifted left,” earlier into the design and coding processes and into the automated test cycles, instead of waiting and running security checks just before release. Security has to fit into the way that engineers think and work: more iterative and incremental, and automated in simple ways.

为了跟上持续交付的步伐，安全必须前移，在代码的设计和测试阶段以及自动化测试阶段，而不是在发布之前去做安全检查。

安全应该溶于工程师的想和做：更多的互动，以及用简单的自动化

Some organizations do this by embedding InfoSec specialists into development and operations teams. But it is difficult to scale this way, because there are too few InfoSec engineers to go around—especially ones who can work at the design and code level. This means developers and operators need to be given more responsibility

一些组织会把信息安全工程师内嵌在开发和运维团队。但是这个不够广。因为很少有信息安全工程师能达到设计的代码级别。这意味着开发人员和运维人员需要去承担更多的责任。

for security, training in security principles and practices, and tools to help them build and run secure systems.

对于安全来说，培训安全原理和实践，以及工具去帮助他们建立和运维安全系统。

Developers need to learn how to identify and mitigate security risks in design through threat modeling, and how to leverage security features in their application frameworks and security libraries to prevent common security vulnerabilities like injection attacks. The OWASP and SAFECode communities provide a lot of useful, free tools and frameworks and guidance to help developers with under‐standing and solving common application security problems in any kind of system.

在设计阶段，开发人员需要通过威胁模型去学会识别和减少安全风险，以及在他们的程序框架中去发挥安全特性的杠杠作用，用安全特征库去阻止常见的安全弱点如注入攻击。

OWASP 和ASFECode组织提供了很多有用和免费的工具以及框架和指导去帮助开发者去抵御和解决常见的程序安全问题。

Making smaller changes in DevOps not only reduces the operational risk of failure; it also reduces security risks, because most small,incremental changes do not meaningfully increase the system’s attack surface. But a red flag should be raised whenever anyone makes a high-risk change, such as changing crypto code or the configuration of a public network–facing device. This can be done automatically on check-in. For example, at Etsy, they has high-risk code and automatically run unit tests as part of Continuous Integration that will alert InfoSec if any of this code changes.

DevOps 倡导 做小的变更不仅仅能降低运维风险；而且能降低安全风险，因为小的、渐进式的变更在增加系统攻击面是没有意义的。当有人做了一个高风险变更，一个红色的标志就会出现。

比如有人修改了密码库代码或者修改面向互联网的公共设备。这个在接入时将会被自动检查。例如，在Etsy,他们高风险代码在自动的运行，在持续集成的测试部分将会给信息安全工程师一个告警信息当有代码变更时。

Self-Service Automated Security Scanning

If you want to make developers more responsible for application security, you need to give them simple tools that work iteratively and incrementally, and that provide fast and simple feedback.

如果你想开发人员对程序的安全更加负责任，那么你就应该给他们简单的工具，这样工作中逐渐的相互交互，然后就能提供快速和简单的反馈。

Scanning applications for security vulnerabilities using automated tools is fundamental to most security programs today. But rather than relying on a centralized security scanning factory run by Info‐Sec, DevOps organizations like Twitter and Netflix implement self service security tools for developers.

用自动化工具去扫描程序安全漏洞在当今是个主流方式。和依赖于集中安全扫描不同的是，推特和奈飞为开发者提供自服务安全工具

While Dynamic Analysis Security Testing (DAST) tools and services are important in testing web and mobile apps, they don’t play that nicely in Continuous Integration or Continuous Delivery.

动态分析安全测试工具和服务在测试页面和移动app方面非常的重要，但是他们在持续集成和持续交付方面并不友好。

Most of these tools are designed to be run by security analysts or pen testers, not a Continuous Integration server like Jenkins or Bamboo. While you can run an attack proxy like OWASP’s ZAP in headless mode to automatically scan a web app for common vulnerabilities, it’s difficult to set them up so that they always return unambiguous pass/fail results to your automated pipeline.

大多数工具都设计给安全分析员或者专业的测试人员使用，而不是给一个持续集成的服务器如Jenkins或者Bamboo去使用。当你去运行一个攻击代理如OWASP的ZAP工具，在无脑模式下，自动的去扫描

一个Web 应用，它很难去实现这个目标，因为在你的自动化工作流中它经常返回模糊的成功或者失败结果。

And more importantly, these tools can’t be used to test system-to-system APIs.This means that Static Analysis Security Testing (SAST) becomes the scanning technology of choice in Continuous Delivery for many financial applications.

更重要的是，这些工具不能别用于系统之间的API 之间。这意味着在大多数金融程序中，在持续交付中采用的是静态分析安全测试

Developers can take advantage of IDE plugins like Find Security Bugs, Cigital’s SecureAssist, or Puma Scan, or plug-ins from Coverity, Klocwork, HPE Fortify, or Checkmarx, to catch security problems and common coding mistakes as they are writing code.

当他们编写代码时，开发人员可以利用编译软件的插件如发现安全漏洞，Cigital的安全助手，或者Puma 扫描，或者来自Coverity,Klocwork,HPE Fortify,或者Checkmarx这些组织提供的插件去找到安全问题以及常见代码错误。

Incremental static analysis pre-commit and commit checks can also be wired into Continuous Integration to catch common mistakes and anti-patterns quickly (full scans, which can take several hours to run on a large code base, need to be run separately, outside of the pipeline). It’s important to tune these tools to minimize false positives, in order to provide developers with clear, actionable, pass/fail feedback.

在提交前和提交后，逐步增加静态分析能够更好的溶于到持续集成中，去更快的发现常见错误和反面模式（全扫描一个大型代码库，需要很长时间，这个需要在流水线外独立的去运行）

调优这些工具去减少误报，以便更好的给开发人员提供清晰，可操作的成功和失败的反馈。

Wiring Security Tests into CI/CD

将安全测试溶于持续集成和持续交付

Scanning code for common security vulnerabilities and coding mistakes isn’t enough. Developers need to include security testing as part of their automated unit and integration tests for security features and functions: positive and negative tests on authentication,access control, and auditing functions and security libraries. Write positive and negative API-level integration tests to check that security functions are working correctly: that you can’t perform an action if you haven’t authenticated, that you can’t see or change information for a different account, and so on.

扫描代码常见安全漏洞和代码错误是远远不够的。开发人员需要安全特性和功能做为自动化单元和集成测试的一部分：正面或者负面的测试在认证、准入控制、审计功能和安全库中。

编写正面和负面的API级别的集成测试去检查安全功能是正确的做法：这样你就不会去做没有认证过的操作，这样你也不会看到或者改变一个不同的账户等等。

Then script system-level attacks in Continuous Delivery using tools that behave well in CI/CD, like Gauntlt, Mittn, or BDD-Security.

在持续集成和持续交付中，系统级别的攻击脚本工具表现非常好，例如：Gauntlt, Mittn, or BDD-Security.

Some common tests that can be done using tools like Gauntlt include using nmap to check for open ports, verifying that SSL is configured correctly, attempting SQL injection and other common attacks, and testing for high-severity vulnerabilities like Heartbleed.

一些常规测试可以用例如Gauntlt 包括使用nmap去检查放开的端口，确定SSL是否正确的配置了，尝试SQL注入以及其他的攻击，并且测试高危风险如心血漏洞

Coming up with good tests takes a good understanding of the application,the runtime environment, and security tools, bringing developers,Ops, and InfoSec together. Like automating integration testing or acceptance testing, it will take a while to build up a strong set of security tests in Continuous Delivery. Start by building a security smoke test: a quick, basic regression test that can be run early in the pipeline and regularly in production to catch common and important security problems, and to ensure that security configurations are always set up correctly.

好的测试需要对程序和生产环境有很好的了解，安全工具将开发人员、运维人员、信息安全人员组织在一起。例如自动化集成测试和可行性测试，它需要在持续交付中建立一个很强的安全测试集。

首先做安全测试：一个快速，简单、回归测试可以在流水线和生产中规则提前运行去发现常见和重要的安全问题，去保证安全的配置被正确的设置了。

Automating security testing makes it easy to collect metrics on the security posture of the application, and to make this information available to everyone—development, Ops, InfoSec, and compliance —as part of the team’s CI/CD dashboard.

自动化安全测试非常容易收集程序的安全画像和尺度，并且让这些信息可以被以下人员使用：开发、运维、信息安全以及合规。 这是持续集成和持续交付面板的一部分。

There is still a place for pen tests and comprehensive security audits in Continuous Delivery, and not just to meet regulatory requirements.

在持续交付过程中，渗透测试和简明的安全审计还有一个地方，它不仅仅是去满足合规的要求。

The real value in a pen test or a security audit is as a health check on the effectiveness of your security practices and controls.

渗透测试和安全审计真正的价值在于它是一个你安全实践和控制的一个有效的健康性检查。

Treat the results the same as a production failure. Run them through a postmortem review to understand the root causes: what you need to improve in your training, reviews, testing, and other checks; what you need to change in your design or coding practices. Just like with a production failure, it’s not enough to fix the problem. You have to make sure to prevent problems from happening again.

把这些结果当做生产故障去对待吧。在事后去运行它们去审视去明白根因：你需要改进你的培训、审视、以及其他检查；

你需要改变你的设计和代码编写。就像对待生产事故一样，仅仅解决问题是不够的，你需要去防止问题的再次发生。

Supply Chain Security: A System Is Only as Secure as the Sum of Its Parts

Today’s Agile and DevOps teams take extensive advantage of open source libraries to reduce development time and costs—which means that they also inherit quality problems and security vulnerabilities from other people’s code.

今天敏捷和开发运维团队利用了大量开源代码库的优势去减低开发时间和开销，这也就意味着他们也继承了其他人代码的质量问题和安全漏洞。

According to Sonatype (who run the Central Repository, the world’s largest repo for open source software), as much as 80% of application code today comes from libraries and frameworks—and a lot of this code has serious problems in it.11

Sonatype(世界上最大的开源软件储存室)发现80%的代码来自库和框架，很多代码中有严重问题。

They looked at 31 billion download requests from 106,000 different organizations in 2015 and found that:

他们发现在2015年，有106,000家不同的机构发起了310亿下载请求：

Enterprises such as large financial services organizations are using an average of 7,600 different software suppliers. These companies sourced an average of 230,000 “software parts” in 2015. One in

every 16 download requests was for a software component which contained at least 1 known security vulnerability.12 More than 50,000 of the software components in the Central Repository

have known security vulnerabilities. In more than half of open source projects, security vulnerabilities are never fixed—even when the project is being actively maintained. Every day, 1,000 new open source projects are created, and 50 new critical vulnerabilities in open source software are reported.

大型的金融服务组织使用了平均7,600家不同的软件供应商。这些企业平均有230,000 软件来自上述下载。

平均每16个下载里面就有一个已知的安全漏洞。在集中储存室中，超过50,000代码组件有已知的安全漏洞。在超过半数以上的开源软件项目中，安全漏洞从来没有被修复。

甚至那些已经有活跃维护的。每天有1000个新的开源项目被创建，有50个新的严重漏洞在这些开源软件中被批露。

Scared yet? You should be. Most organizations have no insight into what components they are using or the risks that they are taking on. You need to know what open source code is included in your apps and when this changes, and you need to review this code for known security vulnerabilities.

难以置信吧？ 你应该相信它。大多数企业没有他们正在使用的组件洞察力以及他们正在承担着怎样的风险。你需要知道的是，这些在你程序中的这些开源代码，当它们改变时，

你需要去审视这些代码是否有已知的安全漏洞。

Luckily, this can be done automatically. Open source tools like OWASP’s Dependency Check, Retire.JS, or Bundler-Audit, and commercial tools like Sonatype Nexus Lifecycle or SourceClear, can be wired into the CI/CD pipeline to detect open source dependencies, identify known security vulnerabilities, and fail the build automatically if serious problems are found.

幸运的是，这些可以自动的去做。开源工具，如OWASP的依赖性检查，Retire.JS, or Bundler-Audit,以及商业软件如Sonatype Nexus Lifecycle or SourceClear，可以融入到持续集成/持续交付的工作流中去检查开源代码的依赖关系，已知的安全漏洞，失败并且自动化编译如果发现了严重问题。

Secure Infrastructure as Code

安全基础即代码

The same ideas and controls need to be followed when making changes to infrastructure. This can easily be done using modern configuration management tools like Puppet, Chef, and Ansible.

当对基础架构做变更时，也可以使用同样的思路和控制。

这个使用现代配置管理工具如Puppet,Chef,Ansible很容易实现。

These tools make it easy to set up standardized configurations across the environment using templates, minimizing the security risk that one unpatched server can be exploited by hackers, as well as the operational risks of a server being set up incorrectly (as we saw in the Knight case study). All the configuration information for the managed environment is visible in a central repository, and under version control.

这些工具通过使用模板很容易做到环境配置文件的标准化，减小了一个没有打补丁的服务器被黑客利用的安全风险，同时减小了因为配置文件没有正确配置的潜在风险(正如我们例子中奈特中看到的那样)。所有的配置文件在集中仓库里都是可视化的，并且是接受版本控制的。

This means that when a vulnerability is reported in a software component like OpenSSL, it is easy to identify which systems need to be patched, and it is easy to push the patch out too.

These tools also provide file integrity monitoring and give you control over configuration drift: they continuously audit runtime configurations to make sure that they match definitions, alert when something is missing or wrong, and automatically correct it.

这也就意味着在软件组件中公布出来的漏洞，如OpenSSL，很容易识别出来那些补丁需要打，也很容易去把补丁推送过去。

Puppet manifests and Chef cookbooks need to be written and reviewed with security in mind. Unit tests for Puppet and Chef should include security checks. Build standard hardening steps into your recipes, instead of relying on scripts or manual checklists.

Puppet手册和Chef指南在编写的时候应该考虑安全。Puppet和Chef的单元测试也应该包括安全检查。 在你指南中应该固化标准步骤，而不要去依赖脚本和手册的检查清单。

There are several examples of Puppet modules and Chef cookbooks available to help harden Linux systems against security guidelines like the Center for Internet Security (CIS) benchmarks and the Defense Information Systems Agency’s Security Technical Implementation Guides (STIG).

在互联网安全中心基准和防护系统信息安全技术实施指导这样的指导下，Puppet单元和Chef手册中有很多例子去帮助加固Linux系统。

Dev-Sec Hardening Framework

安全开发加固框架

The Dev-Sec hardening framework provides a comprehensive set of open source secure configuration templates and automated compliance test suites for Chef, Puppet, Docker, and Ansible that you can use as a starting point for defining and implementing your own hardening policies.

安全开发加固框架为Chef、Puppet、Docker、Ansible提供了详尽的系列开源软件配置模板和自动合规测试套装，你可以把这些作为一个定义和实施你自己的加固策略的起点。

Security Doesn’t End with Development or deployment

安全不应该终止于开发和部署

Another key part of DevOpsSec is tying security into application monitoring and metrics and runtime checks.

另一个关键部分是，开发安全应该烙印到应用监控和测量和运行时的检查中。

Security monitoring in many enterprises is the responsibility of a Security Operations Center (SOC), manned by security analysts who focus on anomalies in network traffic. But security also needs to be tied into application and operations monitoring to identify and catch probes and attacks in context.

在很多企业，都是安全操作中心去做安全监控，安全分析员主要集中精力在网络流量分析上。

但是安全还应该依附于应用和操作的监控 去识别和追踪环境中的攻击行为。

Build instrumentation and intrusion detection into the application using a design framework like OWASP’s AppSensor, and make application attack data and other anomalies visible to operations and developers, as well as to your SOC. This enables what Zane Lackey at Signal Sciences calls “attack-driven defense”: using information on what attackers are doing, or trying to do, in production to understand where you need to focus your security program, and to highlight weaknesses in your systems and controls. These aren’t theoretical problems that you should try to understand and take care of—they are imminent threats to your organization and your customers that must be dealt with immediately.

在程序中使用如OWASP的应用感应器之类的开发框架 去建立仪表板和入侵检测，使得攻击数据和异常对于操作人员和开发人员是可视化的，就像你的SOC一样。这就是Zane Lackey 在

重大科学中所说的”攻击驱动防御”：使用攻击者正在使用的信息，或者尝试去做，你需要在生产中关注安全程序，并且在你系统和控制中高亮显示弱点。这不是假设你需要去明白和处理的问题，这是你的组织和客户真实需要马上去处理的即将发生的危险。

Security runtime checks should also be done as part of application operations. Netflix’s Security Monkey and Conformity Monkey illustrate the kinds of automated continuous checks that can be done in online systems. These are rule-driven services that automatically monitor the runtime environment to detect changes and to ensure that configurations match predefined rules, checking for violations of security policies and common security configuration weaknesses (in the case of Security Monkey) or configurations that deviate from recommended guidelines (Conformity Monkey).

安全运行时的检查应该成为程序运行的一部分。

奈飞的安全钩子和法规钩子展示了各种可以在在线系统自动持续的检查。

这些规则驱动服务自动监控运行环境去检查变动保证配置文件符合预先定义的规则，检查违背安全策略和一般安全配置弱项(如安全钩子)或者偏离推荐的指导的配置（法规钩子）

They run period‐cally online, notifying the engineering teams and InfoSec when something looks wrong.

他们在线定期运行，去提醒工程师团队和安全工程师团队，当他们发现错误时。

While checks like these are particularly important in an engineering-driven environment like Netflix’s where changes are being pushed out directly by engineering teams using self-service

deployment, the same ideas can be extended to any system to make sure that configurations are always correct and safe.

在以工程师为驱动的环境如奈飞，这样的检查尤其重要。

当工程团队使用自服务部署直接推送变更出去，同样的思路也可以延伸到其他系统去保证配置文件是正确和安全的。

DevSecOps at Intuit

研究机构的安全开发操作

Intuit’s security team has played an important role in its successful move to the cloud. When Intuit decided to adopt cloud computing,the security team was the first group to start working with AWS.

在迈向云环境，机构安全团队扮演了一个重要的角色。

当机构决定转向云计算，安全小组是第一个和AWS一起工作的小组。

They took time to experiment and understand how the platform worked, creating a whitelist of approved services and tools for the other teams, and building a set of secure templates, tools, and workflows to help the engineering teams get their jobs done.

他们需要花时间去实验和明白这个平台是怎么工作的，为其他团队创建可允许的服务和工具白名单，建立一组安全模板，工具，工作流去帮助工程师团队完成这些工作。

The security team continuously scans and scores all of Intuit’s systems for security and compliance and publishes a cross-product security scorecard, so that engineering teams, and their VPs, know if and when they are taking on unnecessary risks.

安全团队从安全和合规方面为机构的系统持续的扫描和打分，公布跨产品的安全分数，这样工程师和副总裁们就知道他们正在承受哪些不必要的风险。

When they find security vulnerabilities, the security team pushes them directly into the engineering team’s backlogs in Jira so that they can be prioritized and fixed like other defects.

当他们发现安全弱项，安全团队会直接推送到Jira中，这样他们就优先处理和修复这些缺陷。

Intuit also runs security war game exercises the first day of every week (they call this “Red Team Mondays”).

机构在每个星期的第一天会做攻防演练（他们称这个为红色团队星期一）.

The Red Team, a small group of skilled ethical attackers and forensics specialists, identifies target systems and builds up attack plans throughout the week, and publishes its targets internally each Friday.

红方，是一小群富有经验合法的攻击者和争议的专家，他们确定目标系统，建立全周攻击计划，

然后每周五内部公布他们的目标

The defensive Blue Teams for those systems will often work over the weekend to prepare, and to find and fix vulnerabilities on their own, to make the Red Team’s job harder.

蓝方在周末会全天准备，他们自己会发现和修复漏洞，这样让红方的工作难度增加。

After the Red Team Monday exercises are over, the teams get together to debrief, review the results, and build action plans.

当攻防演练结束后，小组会一起坐下来分析结果，制定行动计划。

And then it starts again.

然后这样的演练反复进行。

This process not only identifies real problems and makes sure that they get fixed, but also exercises Intuit’s incident response and forensics capabilities so that the security team is always prepared to deal with attacks.

这样的流程不仅仅能识别真正的问题，以及确保他们被修复，而且能够锻炼机构的事件响应能力和公关能力，这样的安全团队一直在为处理攻击做准备。

Continuous Delivery (and DevOps) as a Security Advantage

持续交付即是安全优势

A major problem that almost all organizations face is that even when they know that they have a serious security vulnerability in a system, they can’t get the fix out fast enough to stop attackers from exploiting the vulnerability.

所有机构面临的一个重大问题是，即使他们知道他们在系统里面有一个重大漏洞时，他们不能足够快的去阻止攻击者利用这个漏洞攻击。

The longer vulnerabilities are exposed, the more likely it is that the system will be, or has already been, attacked.

漏洞暴露的时间越久，系统很有可能已经被攻击了。

White Hat Security, which provides a service for scanning websites for security vulnerabilities,regularly analyzes and reports on vulnerability data that it collects.

白帽子安全团队，他们提供为Web网页提供扫描安全漏洞，合规分析以及漏洞报表数据服务。

Using data from 2013 and 2014, White Hat found that 35% of finance and insurance websites were “always vulnerable,” meaning that these sites had at least one serious vulnerability exposed every single day of the year.

沿用2013年到2014年的数据，白帽子团队发现有35%的金融企业的Web站点经常有漏洞，这意味着这些网站每天至少有一个严重的漏洞暴露在外面。

Only 25% of finance and insurance sites were vulnerable for less than 30 days of the year.On average, serious vulnerabilities stayed open for 739 days, and only 27% of serious vulnerabilities were fixed at all, because of the costs, risks, and overhead involved in getting patches out.13

只有25%的金融机构的站点每年低于30天没有漏洞，平均，严重的漏洞会存在739天，只有27%的严重漏洞会被彻底修复，因为修复这些漏洞要考虑成本、风险以及经费。

Continuous Delivery, and collaboration between developers, operations, and InfoSec working together in DevOps, can close these vulnerability windows.

在DevOps的持续交付中，开发、运维、信息安全协调工作，可以关闭这些漏洞窗口。

Most security patches are small and don’t take long to code.

大多数安全补丁都比较小，而且不需要太长时间的编码。

A repeatable, automated Continuous Delivery pipeline means that you can figure out and fix a security bug or download a patch from a vendor, test to make sure that it doesn’t introduce a regression, and get it out quickly, with minimal cost and risk.

一个可重复、自动持续交付的流水线意味着你可以指出和修复一个安全错误或者从厂商下载补丁，通过测试区保证不用回退，并且以较小的成本和风险去修复它。

This is in direct contrast to “quick fixes” done under pressure that have resulted in failures in the past.

这个和过去顶着压力去快速修复有很明显的对比。

The Honeymoon Effect

蜜月期的效果

There appears to be another security advantage to moving fast in DevOps.

对于安全来说，快速的进入DevOps还有一个优势。

Recent research shows that smaller, more frequent changes may make systems safer from attackers, through “the Honeymoon Effect”.

最近研究表明，越小的越频繁的变更可能使得系统更加安全，在蜜月期的时候。

Legacy code with known vulnerabilities is a more common and easier point of attack.

带有已知漏洞的遗留代码更容易成为攻击点。

New code that is changed frequently is harder for attackers to follow and understand, and once they understand it, it might change again before they can exploit a vulnerability.

新代码经常意味着增加了攻击者的理解难度，当他理解这段代码时，在他利用这个漏洞攻击前代码已经发生了改变。

Sure, this is a case of “security through obscurity”—a weak defensive position—but it could offer an additional edge to fast-moving organizations.

因此，这是一个很好的安全默默防御的例子，它给企业提供了一个快速接入的墙。

Security Must Be an Enabler, Not a Blocker

安全应该是一个附能者，而不是阻挡者

In DevOps, “security can no longer be a blocker—in places where this is part of the culture, a big change will be needed.”14 Information security needs to be engaged much closer to development and operations,and security needs to become part of development and operations:

在DevOps中，安全不应该在成为企业文化的阻挡者，这需要做一个大的改变。

信息安全需要更紧密的结合开发和运维，而且安全需要成为开发和运维的一部分。

how they think and how they work. This means security has to become more engineering-oriented and less audit-focused, and a lot more collaborative—which is what DevOps is all about.

他们是怎么想的和怎么工作的。这个意味着安全变得对工程师更加开发，更少的从审计角度。

越来越多的合作，这就是DevOps的所有理念。