DevOpsSec: Security as Code

DevOpsSec : 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”。 它们严重依赖于瀑布项目中的安全评审门禁：审视规范和架构，在提交测试之前扫描代码，在投产前进行渗透测试。

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中 ，没有类似瀑布项目的门禁来安排安全审计或者渗透测试。DevOps甚至没有敏捷的安全冲刺或者强化冲刺。安全需要植入在开发和运维过程中，以及在持续交付阶段。

Whether it’s called DevOpsSec or DevSecOps or “Rugged DevOps,”

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

无论是叫做DevOpsSec、DevSecOps或者“坚固的DevOps”或者其它什么名字，

DevOps在安全方面主要是基于以下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、合规和风控团队之间的那些墙，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. 将持续集成和持续交付中的安全测试和安全检查自动化，，包括依赖项的安全检查

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工作流控制，为监管机构提供安全检查的审计跟踪。

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

6. 将安全交织在应用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.

让我们更详细地来看看这些实践，了解需要做些什么。

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，安全必须左移，在代码的设计和测试阶段以及自动化测试阶段去做安全检查，而不是在发布之前。安全应该符合工程师的思维和工作方式：更加迭代化和增量化，并以简单的方式实现自动化。

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。但是这种方式难以规模化实施。因为信息安全工程师太少了，特别是能达到设计和编码级别地。这意味着开发人员和运营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.

对于安全工程师来说，通过安全原理和实践的培训和工具来帮助他们建立和运行安全系统。

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，开发人员需要学习如何通过威胁建模识别和减轻安全风险，以及如何利用他们程序框架和安全库中的安全功能来防止常见的安全漏洞，如注入攻击。OWASP 和SAFECode组织提供了很多有用和免费的工具、框架和指南，以帮助开发者去理解和解决各种系统中常见的应用安全问题。

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中更小的变更不仅仅能降低运维风险，而且能降低安全风险。因为大部分小的、增量的变更并不会显著增加系统的攻击面。但当有人做了一个高风险变更，如更改加密代码或者修改面向公众网络设备的配置时，就应该发出红色标志。

这可以在签入代码时自动完成。例如，在Etsy,他们标识了高风险代码，并且在持续集成时自动运行单元测试。如果有任何这类型的代码变更，将会向信息安全工程师告警。

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.

如果您想要开发人员对应用程序的安全性更加负责，您需要给他们简单的工具，这些工具能迭代和增量地工作，并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应用程序的安全漏洞在当今大多数安全程序的基础。但与依赖于信息安全工程师提供的集中式安全扫描不同，DevOps组织，如推特和奈飞，为开发人员提供自助式安全工具。

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(DAST)工具和服务在测试页面和移动app方面非常的重要，但是它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渗透测试人员使用，而不是给诸如Jenkins或者Bamboo这样的持续集成服务器使用。当您在无头模式(headless)下运行一个攻击代理,如OWASP ZAP,去自动扫描Web 应用的常见漏洞。很难去设置它们，让它们总是将明确的通过/失败结果返回到您的自动化流水线。

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测试系统之间的API。这意味着静态分析安全测试（SAST）成为大多数金融应用程序持续交付的首选扫描技术。

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.

当开发人员编写代码时，他们可以利用IDE的插件，如Find Security Bugs，Cigital的SecureAssist，或者Puma Scan，以及来自Coverity, Klocwork,HPE Fortify,或者Checkmarx的插件，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.

提交前和提交后的增量代码静态分析也可以连接到持续集成中，以快速发现常见错误和反模式（全扫描一个大型代码库可能需要花费几个小时，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/失败的反馈。

Wiring Security Tests into CI/CD

将安全测试连接到CI/CD中

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。开发人员需要将安全特性和功能的安全测试作为自动化单元和集成测试的一部分：对认证、准入控制、审计功能和安全库实施正向和反向的测试。编写正向和反向的API级别的集成测试去检查安全功能工作正常：如您没有认证过就不能去操作，或者您无法查看或者更改其他账户的信息，诸如此类。

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

然后使用在CI/CD中表现良好的工具，如Gauntlt, Mittn 和BDD-Security，在持续交付中将系统级别的攻击脚本化。

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.

一些常规测试可以使用例如Gauntlt等工具完成，包括使用nmap去检查放开的端口，确定SSL是否配置正确的，尝试SQL注入以及其它常规攻击，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.

好的测试需要对应用程序和运行环境和安全工具有很好的了解，将开发人员、运维人员、信息安全人员结合在一起。像自动化集成测试和验收测试一样，在持续交付中建立一个很强的安全测试集需要一端时间。从安全冒烟测试开始：一个快速、基本的回归测试可以在流水线中早期运行，并在生产中定期运行，以发现常见和重要的安全问题，并确保证安全配置始终被正确设置了。

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.

自动化的安全测试可以方便地收集应用程序安全状态的指标，并将这些信息作为团队的CI/CD仪表盘的一部分提供给每个人，如开发、运维、信息安全和合规人员。

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

在持续交付过程中，渗透测试和全面的安全审计还有一席之地，而不仅仅是去满足合规要求。

渗透测试和安全审计真正的价值在于对安全实践和控制的有效性进行健康检查。

把结果视为生产故障。在事后去复盘以了解根本原因：在培训、审视、测试和其他检查中需要改进的内容；您需要在设计或代码实践中更改的内容。就像对待生产事故一样，仅仅解决问题是不够的，您需要确保防止问题再次发生。

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

供应链安全：一个系统的安全是其各个部分的总和

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.

今天的敏捷和DevOps团队充分利用开源代码库来减少开发时间和成本，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

据Sonatype(其运行的中央存储库是世界上最大的开源软件库)称，今天高达80%的代码来自库和框架，其中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:

他们分析了2015年106,000家不同的机构发起的310亿下载请求，发现：

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

*机构等企业平均在2015年使用了万个”部件”平均每16个下载请求中就有一个是针对包含至少1个已知安全漏洞的软件组件。*

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.

在中央库中，超过5万个代码组件存在已知的安全漏洞。在超过半数以上的开源软件项目中，即使是那些得到积极维护的项目，安全漏洞从来没有被修复。每天有1千个新的开源项目被创建，并报告50个新的开源软件严重漏洞。

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？ 您应该是。大多数企业无法洞察他们正在使用的组件或他们正在承担着怎样的风险。您需要知道您的应用程序中包含了什么开源代码，以及此代码合适更改，并且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，这些可以自动完成。开源工具，如OWASP的依赖检查工具，Retire.JS和 Bundler-Audit,以及商业软件如Sonatype Nexus Lifecycle 或 SourceClear，可以融入到持续集成/持续交付的流水线中去检查开源代码的依赖关系，定位已知的安全漏洞，如果发现了严重问题，将使自动化构建失败。

Secure Infrastructure as Code

安全基础设施即代码

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，从而最大限度降低了一个没有打补丁的服务器被黑客利用的安全风险，以及服务器设置错误的操作风险(正如我们在骑士资本的案例中看到的那样)。托管环境的所有配置信息在中央仓库里都可见，并且接受版本控制。

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.

这也就意味着当OpenSSL等软件组件中报告漏洞时，如很容易识别出来那些系统需要修补打，也很容易把补丁推送过去。这些工具还提供文件完整性监控，并让您控制配置漂移：他们不断地审计运行时配置，以确保它们符合定义，在某些东西丢失或出错时发出警报，并自动

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指南在编写和评审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。 在指南中应该固化标准步骤，而不要去依赖脚本或手动检查表。

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).

Puppet单元和Chef手册中有很多例子去帮助强化Linux系统的安全准则,如互联网安全中心(CIS)基准和国防信息系统局的安全技术实施指南(STIG)。

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.

开发-安全加固框架为Chef、Puppet、Docker、Ansible提供了一套全面的开源安全配置模板和自动合规测试套装，您可以把这些作为定义和实施自己的加固策略的起点。

Security Doesn’t End with Development or deployment

安全不会终止于开发或部署

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

DevOpsSec的另一个关键部分是将安全性与应用程序监控、度量和运行时检查联系起来。

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.

很多企业的安全监控是安全运营中心(SOC)的职责，由关注主要网络流量异常的安全分析师负责管理。但是安全还需要与应用程序和操作监控联系起来，以识别和捕获上下文中的探测和攻击。

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.

使用如OWASP的AppSensor之类的设计框架在应用程序中去建立仪表板和入侵检测，使得应用程序攻击数据和异常对于操作人员和开发人员以及您的SOC可见化。这使得Signal Sciences 公司的Zane Lackey称之为”攻击驱动防御”：使用攻击者在生产系统中正在做或者试图去做什么的信息来了解您需要在将安全程序集中在哪里，并突出系统和控制中的弱点。这不是需要您去试图理解和处理的理论问题，是对您的组织和客户迫在眉睫的威胁，必须立即处理。

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).

安全运行时检查也应该成为应用程序运维的一部分。奈飞的安全猴(Security Monkey)和法规猴(Conformity Monkey)展示了可以在在线系统中进行自动持续的检查。

这些规则驱动服务，自动监控运行时环境以检查变动，并保证配置与预定义规则匹配，检查是否违背安全策略和常见安全配置弱点(在安全猴的情况下)或与建议的准则不符的配置（法规猴）。

他们在线定期运行，当出现问题时通知工程团队和信息安全团队。

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.

在像奈飞这样工程驱动的环境中，这样的检查尤其重要。当工程团队使用自助部署直接推送变更，同样的思路也可以延伸到其他任何系统，以保证配置始终正确和安全。

DevSecOps at Intuit

Intuit公司的DevSecOps

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Intuit公司迈向云的过程中，其安全团队扮演了一个重要的角色。当Intuit决定转向云计算时，安全小组是第一个开始使用AWS的团队。

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.

他们花时间来试验和理解平台是如何工作的，为其他团队创建一个已批准的服务和工具的白名单，并构建立一组安全的模板、工具和工作流来帮助工程师团队完成他们的工作。

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.

安全团队不断扫描和打分所有Intuit系统的安全性和合规性，公布跨产品的安全计分卡，以便工程团队和他们的副总裁们知道他们是否和合适承受不必要的风险。

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.

当他们发现安全漏洞时，安全团队会直接推送到工程团队Jira的代办事项中，以便像其它缺陷一样对它们进行优先级排序和修复。

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

Intuit还在每个星期的第一天做攻防演练（他们称这个为“红队星期一”）.

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，是一小群富有经验的白帽黑客和取证专家组成，他们确定目标系统，并在整个星期内建立攻击计划，然后在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.

这些系统的蓝方防御团队通常会在周末进行准备，并自行发现和修复漏洞，使红方的工作更加困难。

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

当“红队周一”攻防演练结束后，各小组会一起坐下来听取汇报，审查结果，并制定行动计划。

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Intuit的事件响应和取证能力，以便安全团队随时准备应对攻击。

Continuous Delivery (and DevOps) as a Security Advantage

持续交付（和DevOps）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.

几乎所有组织面临的一个重大问题是，即使他们知道系统里面有严重安全漏洞，他们也无法以足够快的速度得到修复，去阻止攻击者利用该漏洞。

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，系统受到攻击或者已经受到攻击可能性就越大。

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，为网站提供安全漏洞扫描服务，定期分析和报告其收集的漏洞数据。

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.

根据2013年和2014年的数据，白帽子团队发现有35%的金融和保险网站“总是容易受到攻击”，这意味着这些网站一年中至少每天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%的金融和保险公司的网站在一年中易受到攻击的时间少于30天。平均而言，严重漏洞会存在739天，而且因为修复这些漏洞的成本、风险以及开销，只有27%的严重漏洞会被彻底修复。13

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

在DevOps中，通过持续交付，开发、运维、信息安全之间的协作，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，不需要太长的编码时间。

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.

一个可重复的、自动化的持续交付流水线意味着，您可以找出和修复安全缺陷，或者从厂商处下载补丁，通过测试以保证不会引入回归缺陷，并且以最小的成本和风险将其快速修复。

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

这个和过去在压力下“快速修复”但导致失败的情形形成了鲜明的对比。

The Honeymoon Effect

蜜月效应

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

在DevOps中快速移动似乎还有另外一个安全优势。

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

最近的研究表明，通过“蜜月效应”，更小、更频繁的变更可能使得系统更加安全地免受攻击。

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.

频繁更改的新代码对于攻击者来说很难跟踪和理解，当他理解这段代码时，在他利用这个漏洞攻击前代码可能已经发生了改变。

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

当然，这是一个很好的“默默无闻的安全”的案例-一个薄弱的防御阵地，但它可以为快速行动的企业提供额外的优势。

Security Must Be an Enabler, Not a Blocker

安全应该是一个赋能者，而不是拦路虎

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，“安全不再是一个拦路虎，并成为文化的一部分，这需要一个巨大的改变。14

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.

他们的想法和工作方式。这个意味着安全变得更加面向工程，更少关注审计，并且更加协作，这就是DevOps所要做的。