ThoughtWorks®

TECHNOLOGY RADAR *APRIL '16*

Our thoughts on the technology and trends that are shaping the future

技术雷达官网

微信公众号

thoughtworks.com/radar

最新动态

本版精彩集锦

开源软件，进入良性循环的副产品

在技术雷达中，有些最有影响力的软件来自那些并不以创建软件工具为初衷的公司。比如Facebook，它并不是传统的软件开发工具创造者，却贡献了很多雷达条目。与过去不同，如今越来越多的公司将其重要的软件资产开源，以吸引应聘者和实现自身价值。这创建了一个良性的反馈环：创新的开源产品吸引了优秀的开发者，他们反过来贡献了更多的创新理念。作为副产品，这些公司的框架和库成为业内最流行的产物。这表明软件开发生态系统正在发生巨变，并且进一步证明了开源软件的力量（前提是在恰当的条件下，我们对于Web Scale Envy的建议仍然成立）。

PAAS解惑

很多大型机构把云计算和平台即服务（PaaS）看作一种标准化基础设施、简化部署和运营、提高开发人员生产力的显而易见的方法。但此言尚早，PaaS的定义仍然模糊不清，很多PaaS方法仍然不完整或受到不成熟的框架和工具的影响。一些PaaS解决方案让原本在设施即服务（IaaS）上很容易的事情变得复杂，比如使用自定义的服务定位器(Service Locator)或复杂的网络拓扑，而大家也还在讨论“容器服务”是否能在拥有更多灵活性的前提下提供类似的服务。我们看到很多公司在使用现成的或者逐步建立自己的PaaS，并取得了不同程度的成功。我们认为，现在的PaaS并不是最终态，它只是进化之路上的一个阶段。企业向云和PaaS迁移带来了很多好处，但同时也面临着许多困难和挑战，特别是在整体流水线设计和工具使用方面。技术使用者需要寻找表明“黄金时代来临”的拐点，同时也要避免在实施具体PaaS时的耦合问题。

DOCKER, DOCKER, DOCKER!

容器技术，特别是Docker，已经被证实是一种有效的应用管理技术。它方便了不同环境的应用程序部署，解决了“在这里正常工作，但在别的环境不行”这类问题。我们已经看到了使用Docker的热潮，以及特别是围绕Docker的生态圈的形成，这使得Docker的应用已经超出了开发/测试环境而进入了生产环境。Docker容器已经被用作许多PaaS平台上的“伸缩单元”以及“数据中心OS”平台，这更加速了Docker发展的势头。容器技术在开发和生产环境的推广将引来更多的关注，包括它带来的连锁反应以及它的负面影响。

过度响应式

响应式编程正在变得非常流行，它是一种让组件响应传播进来的数据变化的编程方法，这跟命令式编程大不相同。几乎所有编程语言都拥有自己的响应式扩展。很多生态系统正在逐步支持这种编程范式，特别是用户接口，一般都会用响应式的风格编写。虽然我们喜欢这种范式，但是过度使用基于事件的系统，会导致程序逻辑变得复杂，也使响应式编程变得难以理解，所以开发人员应该更慎重地使用这种编程风格。因为响应式编程非常流行，所以我们也在技术雷达中加入了大量的响应式框架和工具。

贡献者

ThoughtWorks技术顾问委员会由以下人员组成：

Rebecca Parsons (首席技术官)
Martin Fowler(首席科学家)
Anne J Simmons
Badri Janakiraman
Brain Leke
Dave Elliman
Erik Doernenburg
Evan Bottcher
Fausto de la Torre
徐昊
Ian Cartwright
James Lewis
Jonny LeRoy
Mike Mason
Neal Ford
Rachel Laycock
Sam Newman
Scott Shaw
Srihari Srinivasan
Thiyagu Palanisamy

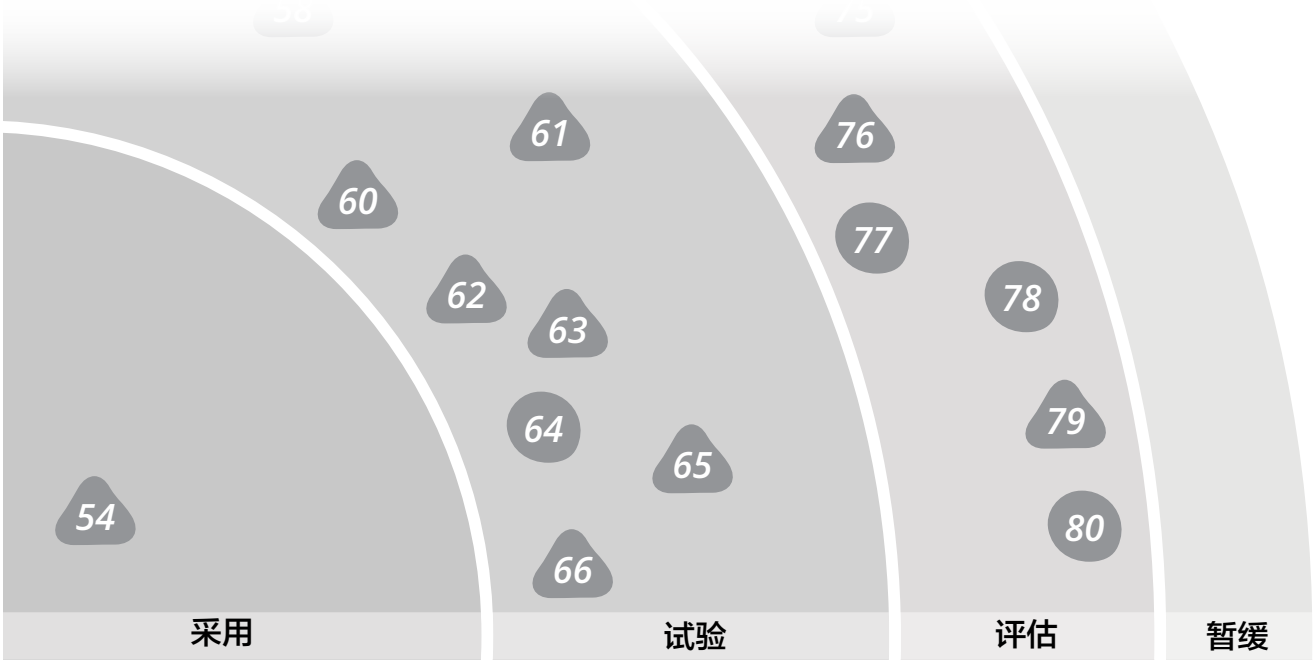
本期技术雷达中文译者：

崔鹏飞
代俊锋
范文博
付莹
官勤
韩锴
黄勇
贾朝阳
李栋
刘杰
刘尚奇
刘宇峰
刘先宁
马博文
苗士博
王萌
王晓峰
王妮
王健
武昆
文浩
伍斌
银大伟
杨政权
姚琪琳
于晓强
钟健鑫
禚娴静
曾磊
张瑞民
张凯峰
周哲武

关于技术雷达

ThoughtWorks 人酷爱技术。我们对技术进行构建、研究、测试、开源、描写，并持之以恒地推动技术的发展——以求造福大众。我们的使命是追求卓越软件并掀起 IT 革命。我们创建并分享 ThoughtWorks 技术雷达，正是为了达成这一使命。ThoughtWorks技术顾问委员会由ThoughtWorks 一群资深的技术领导者组成，他们定期召集会议讨论 ThoughtWorks 的全球技术战略，分析对行业产生重大影响的技术趋势，从而创建了技术雷达。

雷达以独特的形式记录技术顾问委员会的讨论结果，从CIO到开发人员， 雷达为各方利益相关者提供价值。这些内容只是简要的总结，我们建议您探究这些技术以了解更多细节。雷达的本质是采用图形化方式将各种技术归类为技术、工具、平台和语言及框架四个象限。倘若雷达中的某种技术可以被归到多个象限中，我们会选择看起来最合适的一个。我们还进一步将这些技术分为四个环中，由此反映我们目前对它们持有的态度。这四个环分别为：



采用 我们强烈主张业界采用这些技术。如果适合我们的项目，我们会采用它们。

试验 值得追求。重要的是理解如何建立这种能力。企业应该在风险可控的项目中尝试该项技术。

评估 为了确认它将如何影响您所在的企业，值得作一番探究。

暂缓 谨慎推行。

自上次雷达发表以来新出现或发生显著变化的技术以三角形表示，而没有变化的技术则以圆形表示。每个象限的详细图表显示各技术发生的移动。我们感兴趣的技术实在太多，远不是如此大小的文档能合理容纳的，因此我们略去了上期雷达中已包含的许多技术，为新技术腾出空间，略去某项技术并不表示我们不再关心它。

要了解关于雷达的更多背景，请参见 thoughtworks.com/radar/faq

THE RADAR

技术

采用
1. Decoupling deployment from release
2. Products over projects
3. Threat Modeling

试验
4. BFF - Backend for frontends
5. Bug bounties
6. Data Lake
7. Event Storming
8. Flux
9. Idempotency filter
10. iFrames for sandboxing
11. NPM for all the things
12. Phoenix Environments
13. QA in production
14. Reactive architectures

评估
15. Content Security Policies new
16. Hosted IDE's
17. Hosting PII data in the EU new
18. Monitoring of invariants
19. OWASP ASVS new
20. Serverless architecture new
21. Unikernels new
22. VR beyond gaming new

暂缓
23. A single CI instance for all teams new
24. Big Data envy new
25. Gitflow
26. High performance envy/web scale envy
27. SAFe™

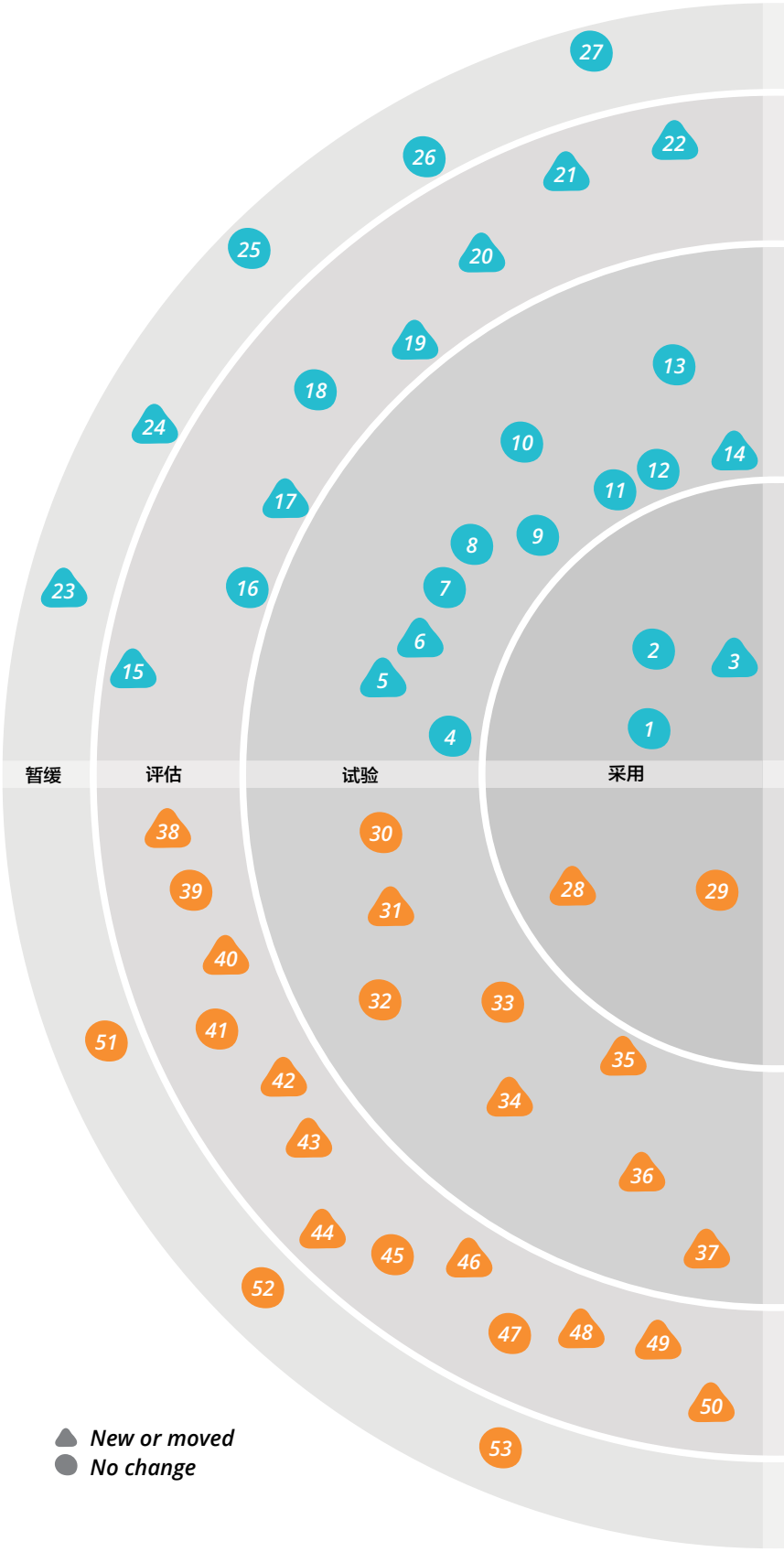
平台

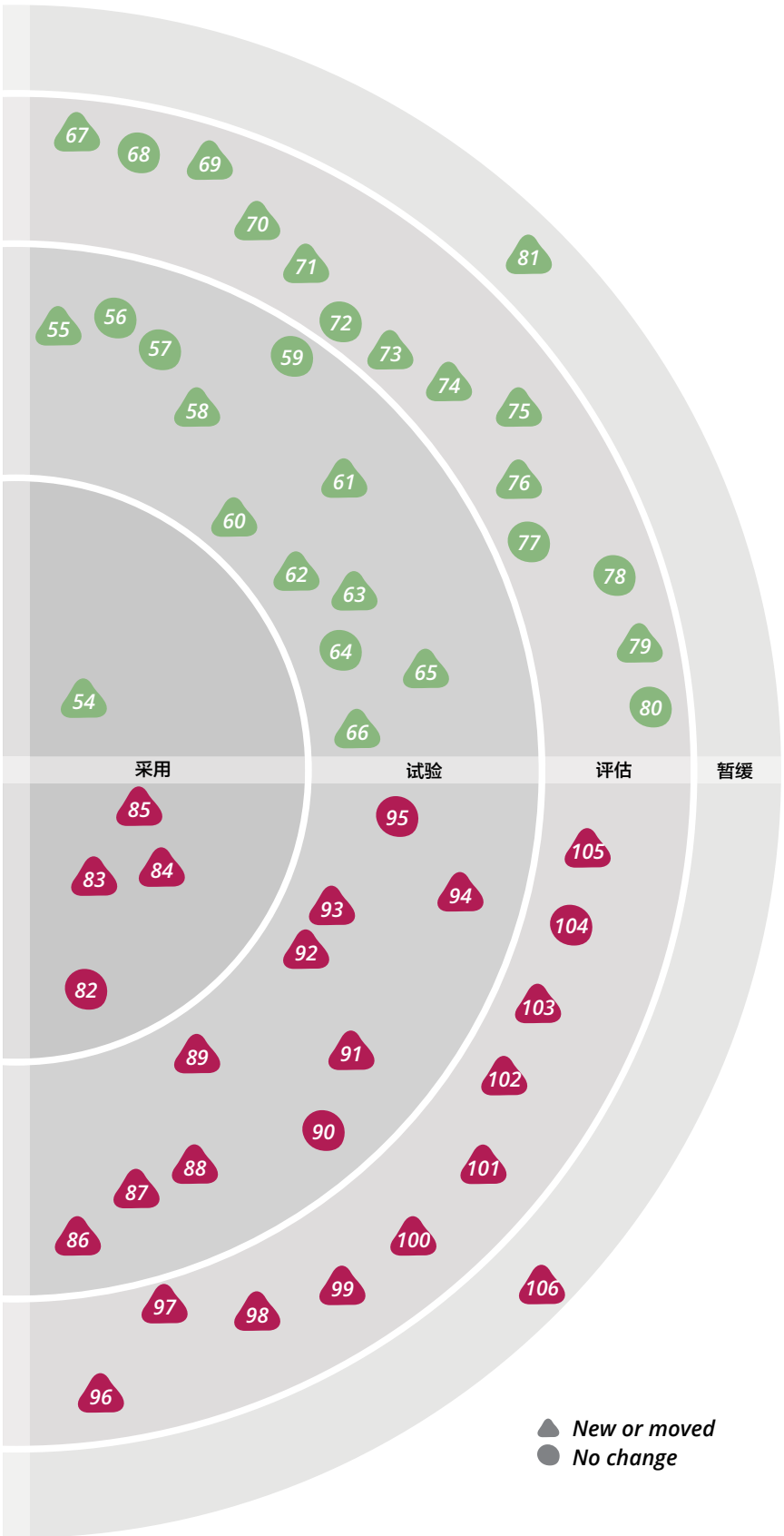
采用
28. Docker
29. TOTP Two-Factor Authentication

试验
30. Apache Mesos
31. AWS Lambda
32. H2O
33. HSTS
34. Kubernetes
35. Linux security modules
36. Pivotal Cloud Foundry new
37. Rancher

评估
38. Amazon API Gateway new
39. AWS ECS
40. Bluetooth Mesh new
41. Ceph
42. Deflect new
43. ESP8266 new
44. MemSQL new
45. Mesosphere DCOS
46. Nomad new
47. Presto
48. Realm new
49. Sandstorm new
50. TensorFlow new

暂缓
51. Application Servers
52. Over-ambitious API Gateways
53. Superficial private cloud





THE RADAR

工具

采用
54. Consul

试验
55. Apache Kafka
56. Browsersync
57. Carthage
58. Gauge
59. GitUp
60. Let's Encrypt
61. Load Impact new
62. OWASP Dependency-Check new
63. Serverspec new
64. SysDig
65. Webpack new
66. Zipkin

评估
67. Apache Flink new
68. Concourse CI
69. Gitrob
70. Grasp new
71. HashiCorp Vault new
72. ievms
73. Jepsen new
74. LambdaCD new
75. Pinpoint new
76. Pitest new
77. Prometheus
78. RAML
79. Repsheet new
80. Sleepy Puppy

暂缓
81. Jenkins as a deployment pipeline new

语言和框架

采用
82. ES6
83. React.js
84. Spring Boot
85. Swift

试验
86. Butterknife new
87. Dagger new
88. Dapper new
89. Ember.js
90. Enlive
91. Fetch new
92. React Native
93. Redux new
94. Robolectric new
95. SignalR

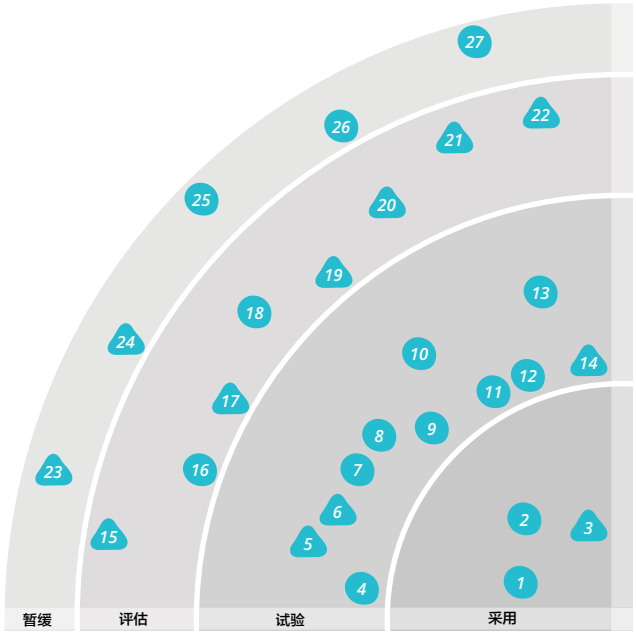
评估
96. Alamofire new
97. AngularJS
98. Aurelia new
99. Cylon.js new
100. Elixir
101. Elm
102. GraphQL new
103. Immutable.js new
104. OkHttp
105. Recharts new

暂缓
106. JSPatch new

技术

随着过去几个月多起著名安全漏洞的公布，软件开发人员毋庸置疑要把编写安全软件和对用户数据负责作为重中之重。但团队不仅面临陡峭的学习曲线，更有数量众多的潜在威胁，从有组织的犯罪和政府间谍，到青少年“随性而为”地攻击系统，都让人头疼不已。**威胁建模**提供了一系列技术，可以帮你早在开发阶段就能够对潜在威胁进行识别和分类。需要指出的是，威胁建模只是安全战略的一部分。当和其他技术（如建立跨职能的安全需求以定位项目所使用技术的常见风险、使用自动化安全扫描器）结合使用时，威胁建模可以变得十分强大。

Bug bounties正被推广到更多企业和政府组织使用。Bug bounty计划鼓励参与者识别潜在的安全漏洞，并向他们提供一些奖品或是认可作为回报。诸如HackerOne和BugCrowd等公司提供服务来帮助这些组织更简单地管理这个过程，而这种服务也正在被更多的组织所接受。



作为数据分析的来源，**数据湖泊**是一种不可变的数据存储，其中存有大量未加工的“原始”数据。尽管这项技术仍然非常容易被错误使用，但我们已经拥有了一些成功案例，因此我们将这项技术移动到“试验”区域。我们推荐使用专门的服务来处理业务系统之间的协作，而把数据湖泊的使用只限制在报表、分析、或给其他数据集市提供数据方面。

当响应式语言扩展和响应式框架大行其道的时候（在这期的雷达中就有几个这样的条目），我们观察到越来越多的团队采用了**响应式架构（reactive architectures）**并取得了成功。尤其是用户界面的设计，借鉴了很多响应式编程的思想。对于这个架构，我们的忠告仍然不变：基于异步消息传输的架构会增加架构的复杂度，系统也会变得更加难以理解，仅仅通过阅读代码已经无法理解系统的功能。在采用响应式风格的架构之前，我们建议先对系统的性能和扩展性需求进行评估。

当处理那些从混合上下文环境中拉取内容的网站时，我们发现**内容安全策略**可以补充我们的安全工具箱。它的工作原理是这样的，即定义一套有关内容合法出处的规则（及是否允许内嵌脚本的标签），浏览器于是可以拒绝加载或执行违反上述规则的JavaScript、CSS或图像。当它与诸如输出编码（output encoding）这样一些良好的实践一同使用时，就能很好地缓解XSS攻击所造成的危害。有趣的是，正是通过用来发送JSON格式违规报告的可选端点，Twitter发现了那些将HTML或JavaScript注入其网页的互联网服务提供商。

从世界上的很多国家，我们都可以看到政府和政府的代理寻求私人，个人身份信息（PII）的广泛访问。在欧盟，最高法

采用
1. Decoupling deployment from release
2. Products over projects
3. Threat Modeling

试验
4. BFF - Backend for frontends
5. Bug bounties
6. Data Lake
7. Event Storming
8. Flux
9. Idempotency filter
10. iFrames for sandboxing
11. NPM for all the things
12. Phoenix Environments
13. QA in production
14. Reactive architectures

评估
15. Content Security Policies
16. Hosted IDE's
17. Hosting PII data in the EU
18. Monitoring of invariants
19. OWASP ASVS
20. Serverless architecture
21. Unikernels
22. VR beyond gaming

暂缓
23. A single CI instance for all teams
24. Big Data envy
25. Gitflow
26. High performance envy/web scale envy
27. SAFe™

技术 接上页

院宣布Safe Harbour框架无效，它的继任者Privacy Shield，也将面临严峻的挑战。与此同时对于云计算平台的使用与日俱增，对于所有的主要云计算提供商，比如 Amazon、Google和Microsoft，它们在欧盟有多个地区和数据中心。因此，我们建议公司，尤其是那些用户覆盖全球的公司，考虑把**PII托管在欧盟范围内**，用欧盟最先进的隐私法律，评估为他们的用户数据提供避风港的可能性。

随着越来越多的开发团队在开发周期早期纳入对安全的考虑，找出限制安全风险的需求就变成令人畏惧的工作。很少有人具备足够多的技术知识去识别应用程序可能面临的所有风险，团队也可能纠结于从何处入手。依赖类似OWASP's **ASVS**（应用程序安全标准）能让这一工作变得更容易。虽然有点冗长，但它包含了一个详尽的需求列表，并按功能分类，比如：认证、访问控制、错误处理、日志记录等等，每个类别都可以按需取用。此外，对测试人员而言它也是验证软件时的有用资源。

无服务器架构致力于使用按需调用的短租计算资源代替一直运行着的虚拟机。在外部请求来到时才获取计算资源，而当请求结束后计算资源立即被释放。这样的应用案例有Firebase以及AWS Lambda。使用这种架构可以减少一些安全问题，例如打安全补丁和SSH权限管理。它还可以更高效利用计算资源。使用这种架构的系统运行成本极低，并且系统本身具有可扩展的特性（尤其是对于AWS Lambda服务更是如此）。举个例子，这个架构可以是一个JavaScript应用，它需要的静态资源放置在CDN或者S3上，动态的AJAX调用则由API Gateway或Lambda负责处理。虽然无服务器架构拥有显著的优点，它也有一些缺点：部署、管理和在服务间共享代码都会变得更加复杂，并且在本地或离线状态下的测试工作会变得更加困难（甚至不可能）。

随着以Docker引领的容器化模型持续升温，我们认为很有必要关注一下持续快速发展的Unikernel领域。Unikernel是精简专属的库操作系统，它能够使用高级语言编译并直接运行在商用云平台虚拟机管理程序之上。相比于容器技术它们有很多的优点，不仅仅是超快的启动时间和更小的攻击面。还有很多技术仍然处于研究阶段，例如微软研发的Drawbridge，MirageOS和HaLVM等，在我们看来这些想法非常有趣，并且能和我们在别处提到的**无服务器架构**技术无缝结合。

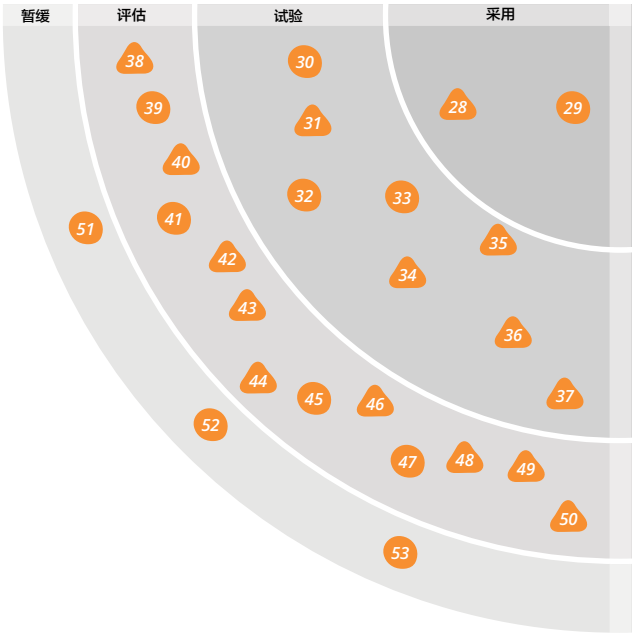
虚拟现实已经有50多年的历史了，伴随着计算技术的持续改进，与之相关的许多想法一直在炒作和探索中。我们相信，现在我们正处在一个转折点。现代显卡能够提供高效的计算能力渲染各种细节，从而呈现高精度分辨率的超现实场景，与此同时至少有两款面向消费者的头戴式设备HTC Vive和Facebook公司的Oculus Rift投入市场。这些头戴式设备价格是很合理的，它们不但具有高清显示屏而且还能够消除可感知的运动轨迹跟踪延迟，这些都是以前提到的造成头痛和眩晕的因素。这些头戴式设备主要面向视频游戏爱好者，但是我们深信它能够不局限于游戏领域，在其他领域会开放更多的可能性，尤其是低端路线，例如Google Cardboard就颇受关注。

虽然印象中，管理一个所有团队共享的**单一CI（持续集成）实例**会更加容易，因为只有一个统一的配置和监控点，但是在一个组织中多个团队共享一个臃肿的CI会导致很多的问题。我们已经发现构建超时、配置冲突和巨型构建队列等类似问题出现得越来越频繁。这种缺陷导致的单点失败会造成多个团队工作的中断。要认真考虑在这些陷阱和保持单点配置之间找到一个平衡点。在具有多团队的组织中，我们建议由各个团队分布式地管理自己独立的CI，这样企业的决策就不会依赖于单个CI实例，但前提是要定义对于它们的选择和配置的标准。

虽然我们早就知道大数据的价值是可以更好地了解用户如何与我们交互，但是我们还是注意到一个惊人的趋势**Big Data envy**，很多组织正在用一些复杂的工具去处理并非是真正的大数据。分布式的map-reduce算法集是一个很便捷处理大量数据集合的技术，但是就我们看来，许多数据集都能够很轻松地用单节点关系数据库和图数据库来处理。即使你真的要处理更多的数据集合，通常要做的第一件事是挑选出你需要的数据，这些数据通常就是用单节点处理也绰绰有余。所以我们想强调的是在你搭建集群之前，你需要以事实出发来评估你需要处理的数据，那么想判断单节点是否适合你，in RAM这个简单的工具可能会给你一些帮助。

平台

我们仍然为**Docker**感到兴奋，因为它逐步从一个工具进化成了复杂的技术平台。开发团队之所以喜爱Docker是因为Docker的镜像格式更容易实现开发和生产环境之间的对等，让部署更加可靠。作为自包含服务的打包机制，它与应用程序的微服务架构风格简直是天作之合。在运维方面，Docker对监控工具（Sensu，Prometheus，CAdvisor 等），编排工具Kubernetes，Marathon等）以及自动化部署工具的支持反映了平台的日益成熟，也同时做好了用于生产环境的准备。尽管Docker和Linux Containers普遍被视为“轻量级的虚拟化”技术，但作为一个忠告，我们并不推荐使用Docker作为安全进程隔离的机制，虽然我们也一直在关注1.10版本中引入的用户命名空间和seccomp（Secure Computing Mode，安全计算模式）。



我们的团队继续享受使用**AWS Lambda**并开始使用它试验无服务器架构，结合Lambda和API网关构建基于隐形基础设施的高伸缩性系统。在使用Java作为Lambda函数启动Lambda容器时，我们碰到了比较大的问题，时不时会有几秒延迟的情况出现，在这种情况下我们建议暂时使用Javascript或是Python进行相关的工作。

如今在分布式集群环境下部署容器的需求场景正变得越来越多，**Kubernetes**正是Google为解决这类问题而推出的容器集群管理框架。实际上Kubernetes并不是一个Google在内部使用的解决方案，而是一个由Google发起并与外部贡献者一起维护的开源项目。自从我们在上次的技术雷达中引入Kubernetes后，我们对其良好的印象得到了进一步确认，并且在我们客户产品环境下看到了成功的应用。

在早期版本的技术雷达中，我们已经强调过**Linux安全模块**的价值，并且讨论了它是如何促进人们思考把加强服务器视作开发流程的一部分。最近，伴随着某些Linux发行版的发行，LXC和Docker容器已经成为默认的Appamor配置，团队必须要理解这些工具是如何工作的。当团队使用容器镜像运行任何不是由他们创建的进程时，这些工具可以帮助团队检查哪些进程访问了共享主机上的什么资源、这些容器内的服务所具备的能力，并且严格控制它们的访问级别。

自从我们在2012版的技术雷达里提到Cloud Foundry后，PaaS领域又有许多新进展。尽管市面上有很多不同的开源核心发行套件，**Pivotal Cloud Foundry**仍然因其丰富的功能和良好的生态圈给我们留下了深刻的印象。尽管我们期望非结构化的方案（如Docker,Mesos, Kubernetes等）与类似Cloud

采用
28. Docker
29. TOTP Two-Factor Authentication

试验
30. Apache Mesos
31. AWS Lambda
32. H2O
33. HSTS
34. Kubernetes
35. Linux security modules
36. Pivotal Cloud Foundry
37. Rancher

评估
38. Amazon API Gateway
39. AWS ECS
40. Bluetooth Mesh
41. Ceph
42. Deflect
43. ESP8266
44. MemSQL
45. Mesosphere DCOS
46. Nomad
47. Presto
48. Realm
49. Sandstorm
50. TensorFlow

暂缓
51. Application Servers
52. Over-ambitious API Gateways
53. Superficial private cloud

平台 接上页

Foundry提供的更结构化和固执的buildpack方案间能趋同共存，我们仍然看到实施PaaS平台为那些愿意接受其限制和演进速度的组织带去了真实的好处。特别是开发团队和平台运维团队交互的简化和标准化带来了开发速度的提升。

新兴的容器即服务（CaaS）如今十分活跃，它提供了介于基础的IaaS（基础设施即服务）和教条式的PaaS（平台即服务）之间的新选择。在保持低调的同时，我们喜欢用简单易用的**Rancher**在生产环境中运行Docker容器。它可作为完整解决方案独立运行，或是与像Kubernetes这样的工具进行集成使用。

Amazon API网关可以帮助开发者轻松创建面向互联网客户的API，并提供了流量管理、监控、认证和授权等常见的API网关功能。我们有不同的团队已经使用此服务作为其他AWS服务（如AWS Lambda）的“前门”来构建无服务器架构系统。我们仍需持续关注那些“雄心勃勃的”API网关所带来的挑战，但当前Amazon API网关服务看起来足够轻量级，避免了那些问题。

尽管大量的智能设备依赖wifi进行部署，我们仍然看到很多成功使用**Bluetooth Mesh**网络的案例，这种网络并不依赖交换机或者网关。Bluetooth LE（低功耗蓝牙）的能耗比wifi更低，同时在智能手机上的接纳度比Zigbee更广。作为一种可自修复的网络，Bluetooth LE用一种很有趣的新手段接入本地设备的区域网络。这种方法已成功应用，不过我们仍期待Bluetooth SIG推出正式的方案。我们乐于见到以更少的基础设施投入来建设一个去中心化的网络，但也不排斥使用网关和云服务的方式，“渐进式增强”现有系统。

Deflect是一款开源服务，帮助非政府组织（NGO）、行动主义者和独立媒体公司免于受到分布式拒绝服务攻击（DDoS）。与商业CDN类似，它基于分布式反向代理缓存，隐藏真实服务器IP地址，同时能够阻止对后台的公开访问，并致力于抵抗针对独立言论的僵尸网络。

日渐增长的硬件黑客群体近来对于**ESP8266**WIFI微控制器感到非常兴奋。与其说是一个具体的技术创新，不如说它在可定制硬件设备的成本和尺寸上，给予了人们更大的发挥空间（可以更便宜和更小）。它主要的特性有：WIFI（能作为基站，访问设备，或二者同时），低功耗，开源硬件，支持Arduino SDK开发，支持Lua开发，庞大的社区支持和与其他IoT模块相比的低成本。

正如摩尔定律预言，我们不断提高计算机系统的能力，并降低其成本，所以使得一些在几年前还看似遥远的处理技术成为可能。内存数据库就是其中一种：不同于将数据存储在缓慢的磁盘或固态硬盘中，我们可以把数据存储到内存中以提高性能。**MemSQL**就是这样一个内存数据库，由于它支持基于集群的纵向扩展，并提供基于SQL的查询语言，MemSQL正在不断产生影响。同时，MemSQL可以与Spark连接进行基于实时数据的数据分析，而不是放任数据在仓库中过时。

Hashicorp还在持续推出有意思的产品—最近引起我们注意的是用于云管理和调度的**Nomad**。在云管理和调度产品竞争激烈的当下，Nomad主要卖点在于（但不限于）容器化的负载和多数据中心/地区的部署。

Realm是一款为移动设备设计的数据库，它通过自身的持久化引擎获得高性能。Realm的市场定位为SQLite和Core Data的替代者，我们的团队用起来感觉很不错。请注意Realm的数据迁移并没有它文档中所宣称的那么直截了当。不过即便如此，Realm还是令我们很激动，建议您加以了解。

对于那些希望使用基于云的协同工具，却不想无意间成为大型云服务提供商的“产品”的人来说，开源的能够自托管的**Sandstorm**成为了一个不错的选择。特别有意思的是其隔离方案，容器化思想应用到了每个文档，而不仅仅是每个应用。而且它还使用了系统调用白名单，以进一步保证沙盒的安全。

谷歌的**TensorFlow**是一个可用于各种从研究到生产的开源机器学习平台，它可在小到一部智能手机、大到大规模图形处理单元（GPU）集群上运行。其重要性表现在，它让实施深度学习算法变得更容易和便捷。抛开那些炒作，其实TensorFlow并不是什么新算法：所有这些技术都已经通过学术界在公开领域存在一段时间了。需要注意的是，对于大多数连最基本的预测分析都还没开始的企业而言，直接进入深度学习其大多数的数据集并没有太大帮助。但是，对于那些有明确问题和数据集的企业，TensorFlow将会是一个有用的工具。

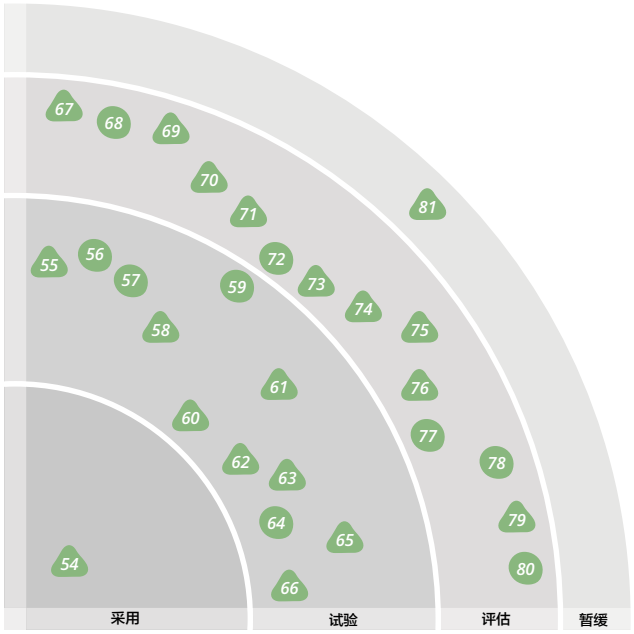
工具

我们将**Consul**——支持基于DNS和HTTP发现机制的服务发现工具——移动到了“采用”区域。它超越了其他的服务发现工具，为注册的服务提供了可定制的健康检查，并且确保不健康的实例能被相应的标记。随着更多协同工具的出现，Consul的功能也更加强大。Consul Template使用Consul提供的数据生成配置文件模板，使诸如在客户端使用mod_proxy做负载均衡变得更容易。在Docker的世界里，registrator使得在docker容器出现在Consul中时，只需要非常小的代价就可以实现自动注册，这让基于容器的配置管理更容易。在使用此类工具之前，你仍然需要从长远角度认真考虑是不是真的需要这种工具，还是采用其他更简单的方法，然而一旦你决定使用服务发现工具，用Consul是一个不错的选择。

现在很多组织都在积极关注新的能够大规模捕获信息作为不变的事件序列的数据架构。开源消息框架**Apache Kafka**持续发力，为大量独立的、轻量级的消费端发布有序的事件提供了解决方案。配置Kafka的工作是很繁琐的，但是我们的团队们对这个框架给予了积极的反馈。

Gauge是一个轻量级的跨平台测试自动化工具。技术规格由自由的Markdown语法写成，因此，测试用例可以用业务语言而不是使用通常的 ‘given-when-then’ 这种具有局限性的格式来描述。不同语言和IDE的支持以插件的形式添加到核心实现中，这使得测试人员能够与团队一起使用同样的支持自动完成、重构等功能的IDE。同时，这个ThoughtWorks出品的开源工具天生就能够并行执行所有支持平台的测试。

Let’s Encrypt第一次出现是在上一版技术雷达中。从2015年12月开始，Let’s Encrypt项目从封闭测试阶段转向公开测试阶段，也就是说用户不再需要收到邀请才能使用它了。Let’s Encrypt为那些寻求网站安全的用户提供了一种简单的方式获取和管理证书。Let’s Encrypt也促使安全和隐私前进了一大步，而这一趋势已经随着ThoughtWorks和我们许多使用其进行证书认证的项目开始了。



Load Impact是一款基于SaaS平台的负载测试工具。这款工具可以模拟多达120万并发用户的访问量。它可以通过Chrome插件来录制回放Web交互行为，也可以模拟移动或是桌面用户的网络连接。除此之外，Load Impact还可以生成来自全球10个不同地域的负载。市面上还有其它的按需使用的负载测试工具，比如说我们也很喜欢BlazeMeter，但是我们的团队对LoadImpact抱有浓厚的兴趣。 满世界的各种库和工具让开

采用
54. Consul

试验
55. Apache Kafka
56. Browsersync
57. Carthage
58. Gauge
59. GitUp
60. Let’s Encrypt
61. Load Impact
62. OWASP Dependency-Check
63. Serverspec
64. SysDig
65. Webpack
66. Zipkin

评估
67. Apache Flink
68. Concourse CI
69. Gitrob
70. Grasp
71. HashiCorp Vault
72. ievms
73. Jepsen
74. LambdaCD
75. Pinpoint
76. Pitest
77. Prometheus
78. RAML
79. Repsheet
80. Sleepy Puppy

暂缓
81. Jenkins as a deployment pipeline

工具 接上页

发人员的日子简单了，但另一方面安全的缺陷也开始变得明显起来，并给使用它们的应用程序增加了安全漏洞。**OWASP Dependency-check**是一款自动识别代码中潜在安全问题的工具，它可以检查代码中是否存在任何已知的漏洞，并持续更新现有的漏洞数据库。Dependency-check为Java和.NET（我们已经有成功的应用）以及Ruby、Node.js、Phyton提供了一些接口和插件来自动扫描代码。

之前我们已经推荐了自动化配置测试技术，其中我们指出**Serverspec**是一个实现这类测试的流行工具。Serverspec并不是一个新的技术，随着越来越多的跨功能交付团队承担基础设施配置管理的职责，我们看到对它的使用也变得越来越普遍。Serverspec基于ruby RSpec，它提供了一套完善的工具类来判断服务器配置的正确性。

Webpack已经证明了自己是值得选择的JavaScript模块打包工具。伴随着这份加载器列表的不断增长，Webpack可以为你所有的静态资源提供单一的依赖树，对JavaScript、CSS等资源进行灵活的操作，并将向浏览器发送内容的数量和次数最小化。特别重要的是，它可以平滑的在AMD、CommonJS以及ES6模块间进行集成，从而让团队能够使用ES6，并在有浏览器兼容性需要的时候自动无缝转译（通过Babel）为更早版本。我们的很多团队也觉得Browserify值得一用，它做了类似的工作，但是更专注于如何让Node.js模块在客户端变得可用。

在2015年中旬**Zipkin**迁移到了Github的openzipkin/zipkin组织，开发工作仍然在继续进行。目前已经有基于Python, Go, Java, Ruby, Scala和C#的API客户端，那些想快速上手的读者也可以使用Docker镜像来搭建Zipkin环境。Zipkin的社区非常活跃，也一直在发展壮大。它的使用方法也变得越来越简单，基于这些原因我们仍然推荐这个工具。如果你需要端到端地跟踪不同请求的延迟，Zipkin仍然是一个非常不错的选择。

Apache Flink是新一代的可伸缩、分布式批处理和流处理平台。其核心是一个数据流引擎。它还支持表格式（像SQL）、图形式的数据处理和机器学习的操作。Apache Flink因其丰富的流处理特性脱颖而出，比如：事件时间、丰富的流窗口操作、容错能力以及一次性（exactly-once）语义。尽管还没有发布1.0版本，但它已经因其对流处理的创新、内存处理、状态管理和简易的配置而备受社区关注。

攻击者持续使用自动化软件来爬取GitHub上公共的代码库，想发现AWS的登录信息并启动EC2的实例，以达到赚取比特币或其他邪恶的目的。虽然采用像git-crypt和Blackbox这样的工具越来越多，它们可以把密码和访问令牌等保密信息安全地存放在代码库中，但多数情况下，这些保密信息并未得到足够保护。而且项目的保密信息被无意中签入到开发者个人代码库的情况也很常见。**Gitrob**可以帮助把暴露保密信息带来的破坏降到最低。它可以扫描一个组织的GitHub代码库，把所有可能包含敏感信息的，不应该被放在代码库的文件标识出来。该工具的当前版本有一些限制：它只能用来扫描公开的GitHub组织以及相应成员，它不会检查文件的内容，也不会检查整个提交的历史，而且每次运行都会完整扫描所有的代码库。尽管这样，它仍然是一个有帮助的反应式工具，能在追悔莫及之前警告团队。相对于Talisman这样的主动式工具来说，它应该是一种补充。

Grasp这个小小的JavaScript的命令行重构工具让我们所有人印象深刻。它为抽象语法树提供了丰富的选择器和操作，比摆弄sed和grep要先进多了。这给我们正在进行的将JavaScript做为一等编程语言的运动添加了一个有用的新工具。

用安全的方式去管理项目的机密信息（秘钥，API key，证书等）正在逐渐成为项目的一大问题。特别是在应用需要访问多种机密信息的环境，例如微服务这样拥有多个应用的环境或者微容器的环境中，之前通过文件或者环境变量存储机密信息的方式已经变得难以控制。**HashiCorp Vault**试图通过使用统一接口安全地访问机密信息来解决这个问题。相对于其他工具，它具有一些简单易用的特性，比如为已知的工具和加密方式自动生成secrets等等。

随着NoSql数据存储方式的使用率增加，以及多存储方式混合解决方案日趋流行，开发团队在需要存储数据时有了多种选择。虽然这能带来很多益处，但是在不稳定的网络环境下产品行为有可能出现微妙的（以及不那么微妙的）问题，而这些问题有时即便是产品开发人员自己也难以理解。当任何人试图了解不同数据库和队列技术在不同场景下如何反应时，事实上都会把**Jepsen**及其博客作为参考。关键一点，使用这种在事务中包含客户端的方法来测试，给许多构建微服务的团队揭示了可能的错误模式。

工具 接上页

LambdaCD让团队能够使用Clojure语言定义持续交付流水线。这给持续交付服务器的配置带来了代码化基础设施的种种好处：源代码管理，单元测试，重构和代码重用。在“代码化流水线”这个领域，LambdaCD代表着轻量化、自包含、完全可编程，使得团队可以用处理代码的方式来管理他们的流水线。

使用了凤凰服务器或者凤凰环境技术的团队经常会发现，现有的应用性能管理（APM）工具带来的帮助非常有限。长期、有限数量的授权模式、在处理短期存在的虚拟硬件时的缺陷都意味着这些工具带来的问题要比解决的问题更多。但分布式系统需要被监控，同时很多团队也越来越意识到APM工具的重要性。我们认为APM领域的开源工具**Pinpoint**是一个值得研究的工具，可以使用它作为AppDynamics和Dynatrace的替代品。Pinpoint使用Java语言实现，对于很多操作系统、数据库和框架都有相应的插件支持。同时它也可以和其他的轻量级开源工具结合使用，比如在本期雷达中提到的Zipkin。如果你还在考虑该购买哪一款APM工具，可以先考虑一下Pinpoint。

Pitest是基于突变测试（mutation testing）技术的Java测试覆盖率分析工具。传统的测试覆盖率分析倾向于度量被测试执行的代码行数，但这种方法仅仅能够识别出完全没有被测试到的代码；而突变测试更进一步，它会测试被执行代码的质量，并且试图发现其中可能存在的其他常见错误。通过这种方法，团队能够发现一些问题，从而帮助团队度量和发展出一套更为有效的测试套件。大多数这样的工具运行起来很慢，而且很难用，不过Pitest已经被证明有更好的性能表现，容易在项目中使用，而且处于良好的维护状态。

使用机器人对Web资产的攻击正变得越来越复杂。而**Repsheet**项目的目标正是识别这些攻击者和它们的行为。它可以是Apache或者NGINX的插件，能通过预定义和用户自定义的规则，记录用户活动、指纹，从而采取相应的动作，包括屏蔽恶意攻击者。它包含一个能够可视化当前攻击者的工具：这样就能提高团队管理基于机器人的威胁的能力，增强团队的安全意识。我们很喜欢它，因为这是一个很好的例子——一个简单的工具解决了一个非常真实，但通常被忽视的问题——基于机器人的攻击。

我们知道这是一个危险的领域，因为我们创造了一个竞争的工具，但是我们觉得必须提出这个长期存在的问题。对于软件开发来说，像CruiseControl和Jenkins这样的持续集成工具是非常有价值的，但是当你的构建流程变得更加复杂时，需要的就不仅是持续集成了，而是需要一个部署流水线。我们经常看到人们利用插件将**Jenkins作为部署流水线**，经验告诉我们，这么做很快会陷入麻烦。Jenkins2.0引入了“流水线即代码”，但是它依然在沿用插件的方式来建模流水线，而没能用Jenkins产品的核心去直接进行流水线建模。按照我们的经验，围绕部署流水线作为一等公民去构建工具更加合适，这也是驱动我们用GoCD去替换CruiseControl的原因。今天我们看到了几个拥抱部署流水线的产品，包括ConcourseCI，LambdaCD，Spinnaker，Drone，以及GoCD。

语言和框架

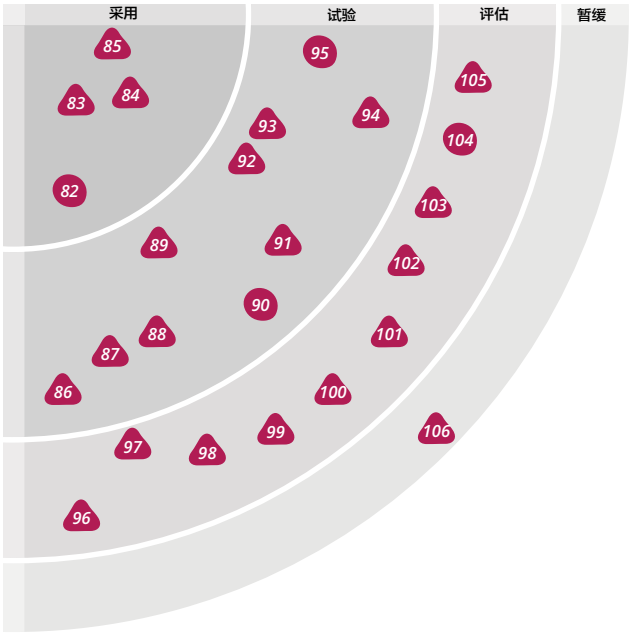
前端JavaScript框架呈现井喷式发展，其中**React.js**凭借其响应式的数据流设计脱颖而出。它只允许单向数据绑定的特性极大的简化了页面渲染逻辑，同时避免了使用其他框架开发应用程序的过程中遇到的诸多问题。我们在越来越多大大小小的项目中感受到React.js带来的好处，与此同时我们将持续关注它和其他未来主流框架（例如AngularJS）的动态。React.js正在成为我们首选的JavaScript框架。

很多的工作已经通过使用**SpringBoot**来降低复杂度和依赖，这在很大程度上缓解了我们以前的保留意见。如果你在Spring的生态系统中并正在走向微服务架构，SpringBoot就是当下最好的选择。而那些不在Sping生态环境中的项目，Dropwizard也值得认真考虑。

Swift现在已经是Apple生态圈中的首选开发语言。从Swift2提供给大多数项目所需要的稳定性和性能来看，它已经接近成熟水平。有大量用于支持iOS开发的类库正在被迁移到Swift，例如：SwiftyJSON, Quick等，其它还没有迁移的应用程序可以参考它们。现在Swift已经被开源，我们也看到开发者社区正在专注于持续提升iOS的开发。

Butterknife是为视图注入绑定字段和方法的类库。它允许注入任意对象、视图和监听器，从而减少Android开发中的胶水代码保持代码整洁。使用Butterknife可以将多个视图组织成列表或数组然后同时应用共同的动作，它并不严重依赖xml配置。我们的项目使用过这个类库，从它的简单和易用性中受益匪浅。

随着对Android应用需求的增加，**Dagger**提供了全静态、编译时依赖注入的框架。Dagger严格生成的实现和不依赖反射的方案解决了很多性能和开发问题，从而使它非常适合Andriod开发。使用Dagger能获得初始和创建的完整调用栈，从而在调试时有完整的可追溯性。



Dapper是基于.NET的极简轻量级ORM框架。它把SQL查询直接映射到了动态对象，而不是试图替你编写SQL查询。虽然它不是新事物，但Dapper已逐步获得了ThoughtWorks里.NET开发团队的认可。对于C＃程序员，它不但避免了一部分关系查询与对象之间映射的苦差事，同时还允许完全控制SQL或存储过程。

Ember.js基于项目实践经验进一步开发支持，成为Javascript应用框架领域的强力竞争者。Ember因为其开发者体验而备受好评，相比之下其它框架（如AngularJS）就少些惊喜了。Ember的命令行构建工具，约定优于配置的设计，以及ES6的支持为它赢得了诸多赞誉。 我们的团队正在从JQuery或者原始的XHR转为使用**Fetch** API和**Fetch** polyfill技术来进行JavaScript远程调用。虽然语义上相似，但是对Promise和

采用

82. ES6
83. React.js
84. Spring Boot
85. Swift

试验

86. Butterknife
87. Dagger
88. Dapper
89. Ember.js
90. Enlive
91. Fetch
92. React Native
93. Redux
94. Robolectric
95. SignalR

评估

96. Alamofire
97. AngularJS
98. Aurelia
99. Cylon.js
100. Elixir
101. Elm
102. GraphQL
103. Immutable.js
104. OkHttp
105. Recharts

暂缓

106. JSPatch

语言和框架 接上页

CORS支持更为简洁。我们认为这将成为一个新的趋势。

我们看到**React Native**在跨平台移动快速开发领域获得了持续的成功。虽然在使用它的过程中会遇到一些问题，但它能够让原生的代码、视图与非原生的代码、视图非常简单地集成，它加快了开发周期（瞬时重新加载、chrome调试、Flexbox布局），同时还有React风格的崛起，最终让我们选择了它。很多框架关心如何让代码保证保持良好的结构，搭配使用类似于Redux的工具会很有帮助。

Redux是一个强大成熟的工具，它在许多项目中帮助我们重新设计如何管理客户端应用程序状态。它使用Flux-style实现了一个松耦合的更容易推断的状态机架构。我们发现，它与其他我们喜爱的JavaScript框架（例如Ember和React）配合使用效果会更好。

在安卓应用开发的世界，**Robolectric**是一个被我们的团队广泛使用的单元测试框架。它为我们提供了最佳选项来通过扩展或是直接使用安卓组件编写真正的单元测试，同时支持junit测试。不过我们依然要提醒Robolectric只是安卓SDK的一个实现，可能某些测试会在个别设备上出现一些问题。手动mock所有的安卓依赖来确保只进行系统级的测试需要大量复杂的代码，而这一框架有效地解决了这个问题。

很多年来，iOS应用程序中的网络和解码部分一直是个麻烦。已经有很多的类库和方案去解决这个由来已久的难题，**Alamofire**是一个非常健壮的对程序员友好的处理JSON解码的类库。它是由Objective-C中类似的库AFNetworking的作者创建，AFNetworking曾经在Objective-C的时代被大量使用。

虽然我们使用**AngularJS**成功交付了很多项目，并且也能看到大型企业中越来越多的项目采用该框架，但是我们决定在这个版本的技术雷达将Angular移回"评估"。这个改动是为了让大家注意：React.js和Ember也有很不错的可选性，Angular从1.0到2.0的迁移过程充满不确定，同时我们发现一些组织在使用这个框架时并没有认真思考单页应用是否适合他们的需要。为此我们进行了激烈的内部辩论，但是可以肯定的是，同时使用双向绑定与不一致状态管理模式会让代码变得过于复杂。另外我们相信，相比于尝试移除一个固有框架，更好的方式是通过仔细的设计，在外层使用Redux或者Flux，来解决这些问题。

Aurelia采用最新的Javascript:ECMAScript 2016标准开发而成，被认为是下一代JavaScript客户端开发框架。Aurelia的作者Rob Eisenberg是Durandal之父，离开Angular2.0核心团队之后全力打造了Aurelia。Aurelia最了不起的是它的高度模块化，包含了许多小型库，可以非常方便的进行定制化开发。Aurelia遵循约定优于配置的理念，而且其约定恰到好处，很容易进行模块的产生和使用。Aurelia有一个庞大的开发社群，它的官网还提供了非常好的入门文档。

IoT设备和Javascript生态系统的融合蕴含着很多有意思的可能性。**Cylon.js**这个Javascript库令技术社区非常兴奋，它能够为机器人和物联网构建接口。它支持50多种平台设备，并由cylon-gpio模块提供的共享驱动来支持通用输入/输出接口。利用这个库，我们就可以通过浏览器来控制设备了。

我们不断的看到许多关于**Elixir**语言的令人兴奋的反馈。Elixir是一款建立在Erlang虚拟机之上的动态编程语言，在构建高并发、高容错性的应用程序方面展现出了优秀的能力。Elixir具有很多与众不同的特性：例如管道操作符，它允许开发人员构建一个类似于UNIX shell命令的函数管道。Elixir通过共享字节代码与Erlang互连，在利用现有库的同时支持Mix构建工具、Iex互动壳层和ExUnit单元测试框架等多种工具。

由于Redux被广泛应用，我们重新讨论了作为Redux设计来源的**Elm**。Elm用一种编译型，强类型的函数式语言的方式来实现了React.js的组件化视图与响应式，同时拥有Redux的可预测化状态。Elm本身用Haskell编写，因此语法与Haskell相似，但是会被编译成HTML，CSS和JavaScript在浏览器端执行。对于将要拥抱React.js和Redux的JavaScript工程师，我们建议也可以考虑将Elm作为一种类型安全的选项。

当我们审视现实世界里的REST实现，经常发现它被错误地当做通过客户端服务器间频繁交互来获取对象图网络的手段。对于这种非常普遍的情况，有一种比REST更合适的方法，那就是Facebook的**GraphQL**，它是一个很有趣的替代方案。GraphQL做为一种远程接收对象图网络的协议，在近期获得了非常高的关注。GraphQL最有趣的特性之一，是它本质上面向消费者，响应体的结构不取决于服务端，而是完全

由客户端驱动。这样它就将消费者解耦，并且强迫服务端遵守Postel法则。目前客户端支持多种编程语言，我们看到人们对Facebook的Relay表现出强烈的兴趣。它是一个支持到React.js无状态组件模型的Javascript框架。

函数式编程范式经常强调不可变性，并且大多数的编程语言原生支持创建不可变对象，他们一旦创建就不能被修改。**Immutable.js**是一个提供很多永久不可变数据结构的JavaScript工具库，这样的数据结构在JavaScript虚拟机中非常高效。Immutable.js对象和原生JavaScript对象是有区别的，所以需要避免直接引用immutable对象中的JavaScript对象。我们的团队借助它让追踪变换和状态维护变得更容易。我们鼓励程序员去研究和使用这个类库，尤其是跟其他Facebook技术栈一起使用的时候。

我们很享受Recharts将D3图表库以很简洁和声明式的方式整合到React.js后的开发体验。

许多iOS开发人员使用**JSPatch**动态地给app打补丁。当一个启用了JSPatch的app运行时，程序会先加载一个JavaScript块（可能通过不安全的HTTP连接），然后与Objective-C应用代码通信建立连接，从而改变代码行为、修改bug。虽然这样很方便，但是在APP中留一个monkey patch并不是一件好事，也不应该这样做。为了能够对功能进行合理地验证，在进行任何增量修改之时，有一个能够与最终用户体验相吻合的测试过程显得非常重要。有另外一种方式是使用React Native构建APP，然后使用AppHub或者CodePush来推送小的更新和功能。
