

DevSecOps的十条建议

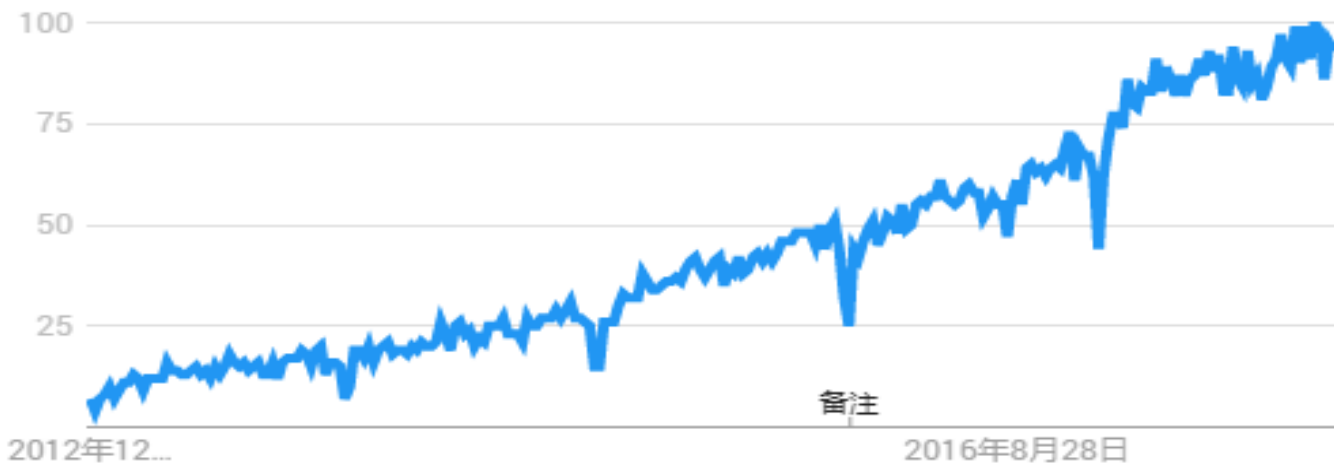
2017.12 上海



热度随时间变化的趋势

Google Trends

● devops



Business Benefits with DevOps

edureka!

 **57%**
INCREASED customer conversion or satisfaction

 **57%**
REDUCED IT Infrastructure spend

 **46%**
INCREASE In sales

 **32%**
INCREASE In employee engagement

 **49%**
REDUCTION In application downtime or failure rates

 **46%**
INCREASE In customer engagement

 **3%**
No, the wider business has not seen measurable benefits from Devops

 **2%**
It's too early to say

IT Benefits with DevOps

edureka!



44% FASTER TIME
to market for new software releases



45% MORE
Innovation



44% IMPROVED
application stability



44% ABILITY
to respond quickly to the
business requirements

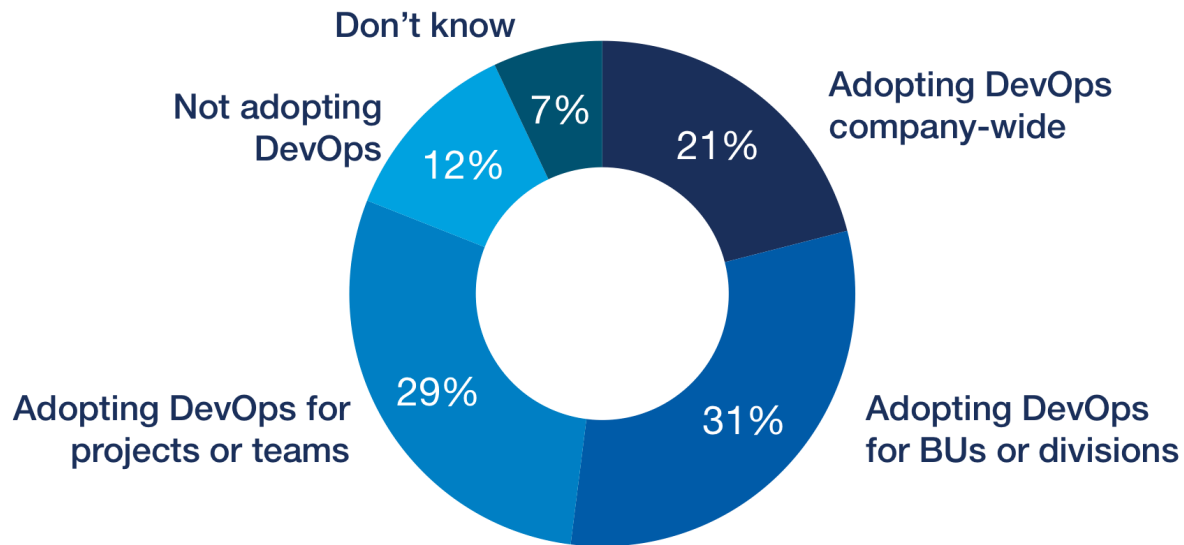


34% REDUCED
IT costs



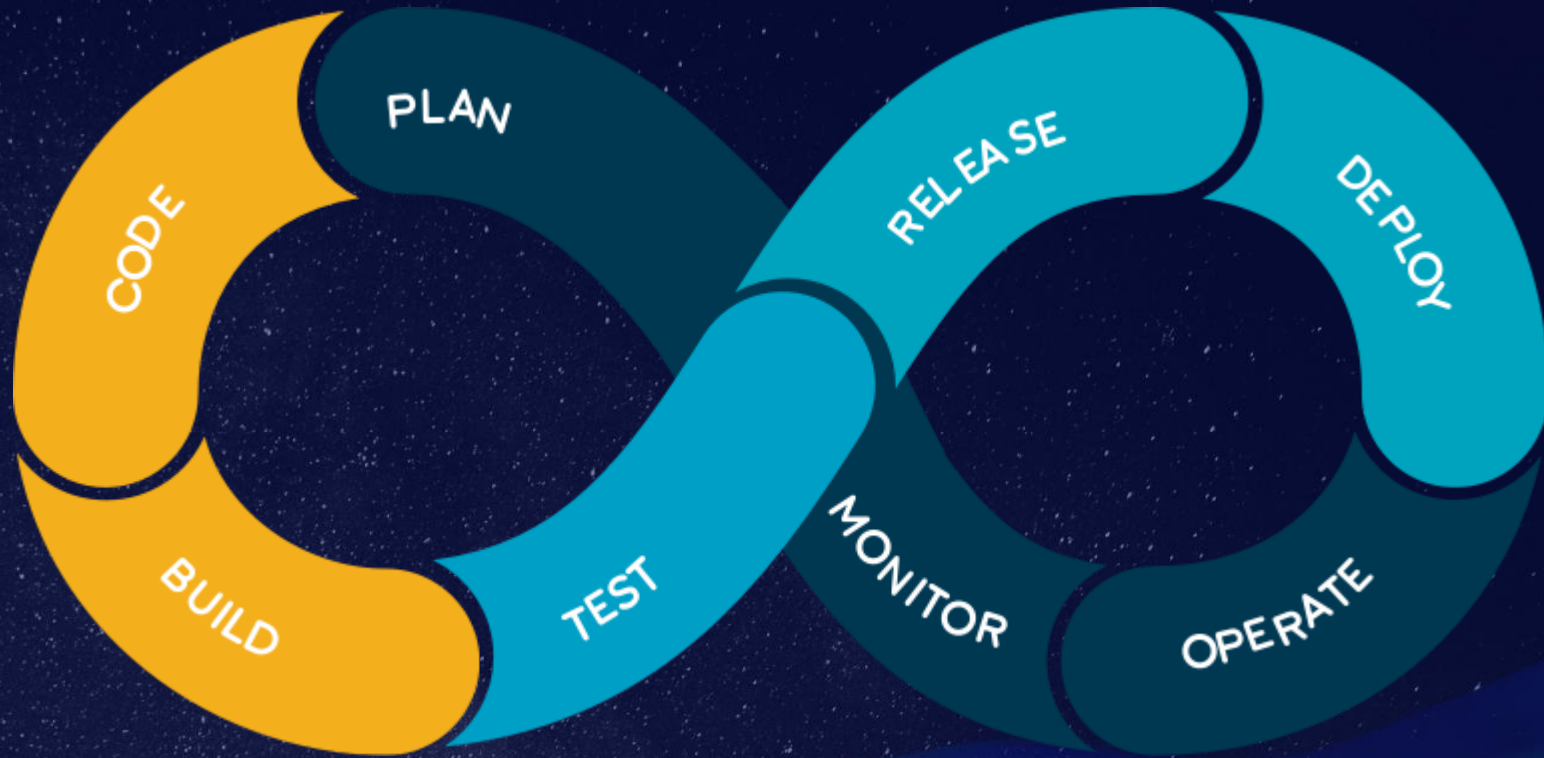
17% THE BUSINESS
RECOGNIZES the value IT can add

Enterprise Adoption of DevOps



Source: RightScale 2016 State of the Cloud Report



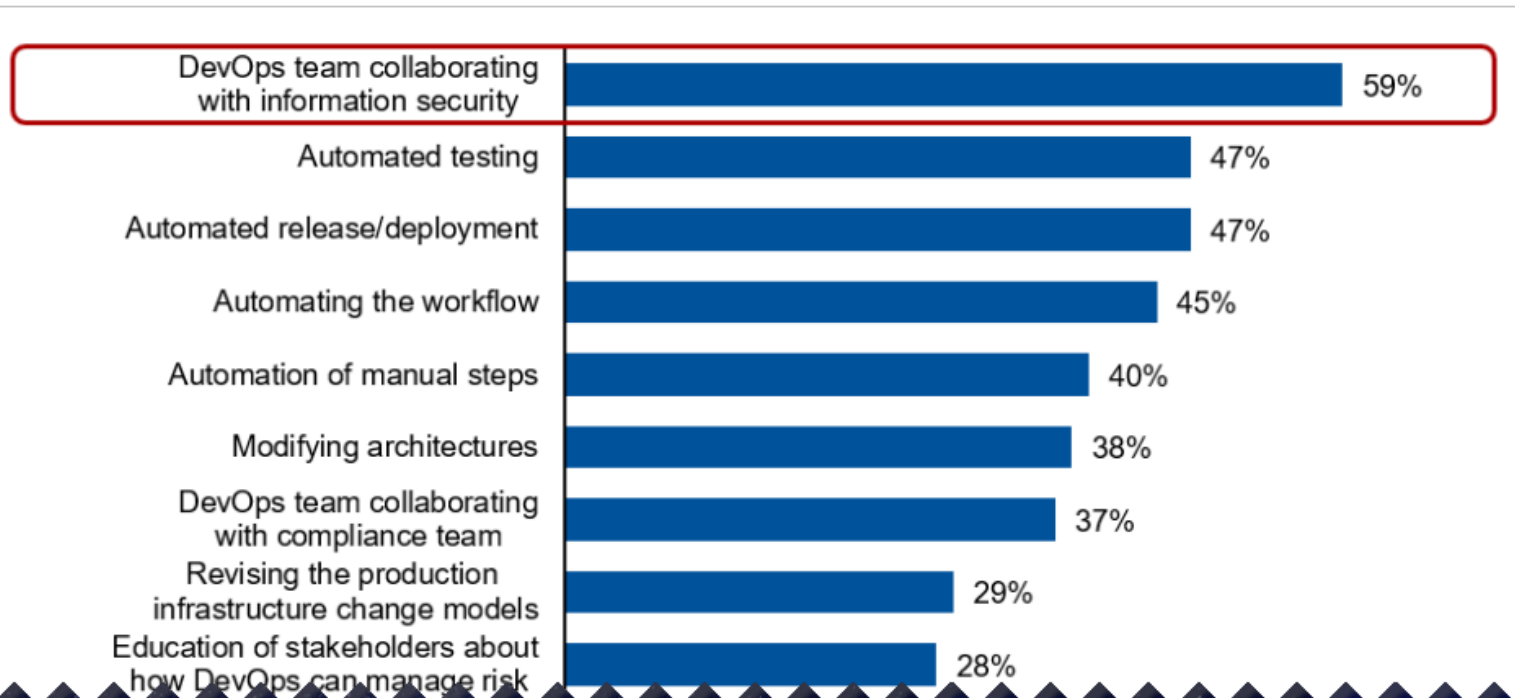


安全在哪？

FTT · REEBUF



Gartner的一份调查显示，59%的DevOps团队希望与安全进行良好的协作



DevOps+Sec



DevSecOps

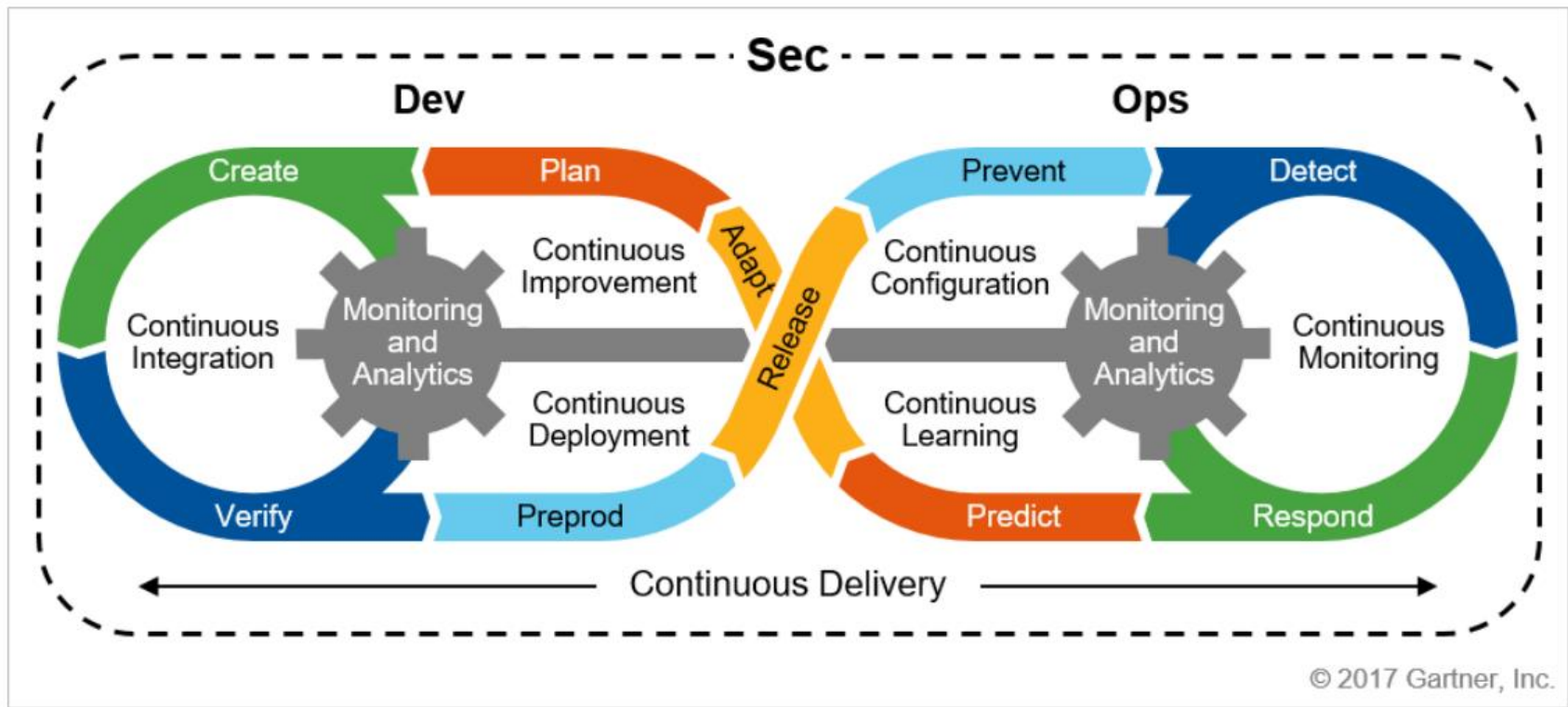
**改变思维，成为服务者和合作者
与开发紧密集成，实现持续交付
抓住主要矛盾和矛盾的主要方面
让事情变得更简单
从头就要安全
努力自动化
赋能**

将安全视为持续交付、学习和改进的过程

DevOps的动机是消除Dev和Ops当中的隔阂，而Sec则是另一个应该消除的隔阂。安全应该克服关卡式思维，与开发和运维进行深度的融合，并贯穿到全生命周期中。

- DevOps是持续的，所以Sec也要持续。完整的DevOps生命周期需要安全加固，包括新的服务部署到已有环境中。将安全理解为通过DevSecOps对开发和运维进行持续改进的流程，安全就象开发一样，将变成持续的交付、学习和改进的流程。
- 要密切关注在各个阶段的安全状况，及时感知存在的问题，并及时反馈给相关的人员，尽快解决。
- 要尽可能地实现自动化和可编程化，Security as Code，这样才能更好地整合到DevOps的流程中

随风潜入夜
润物细无声



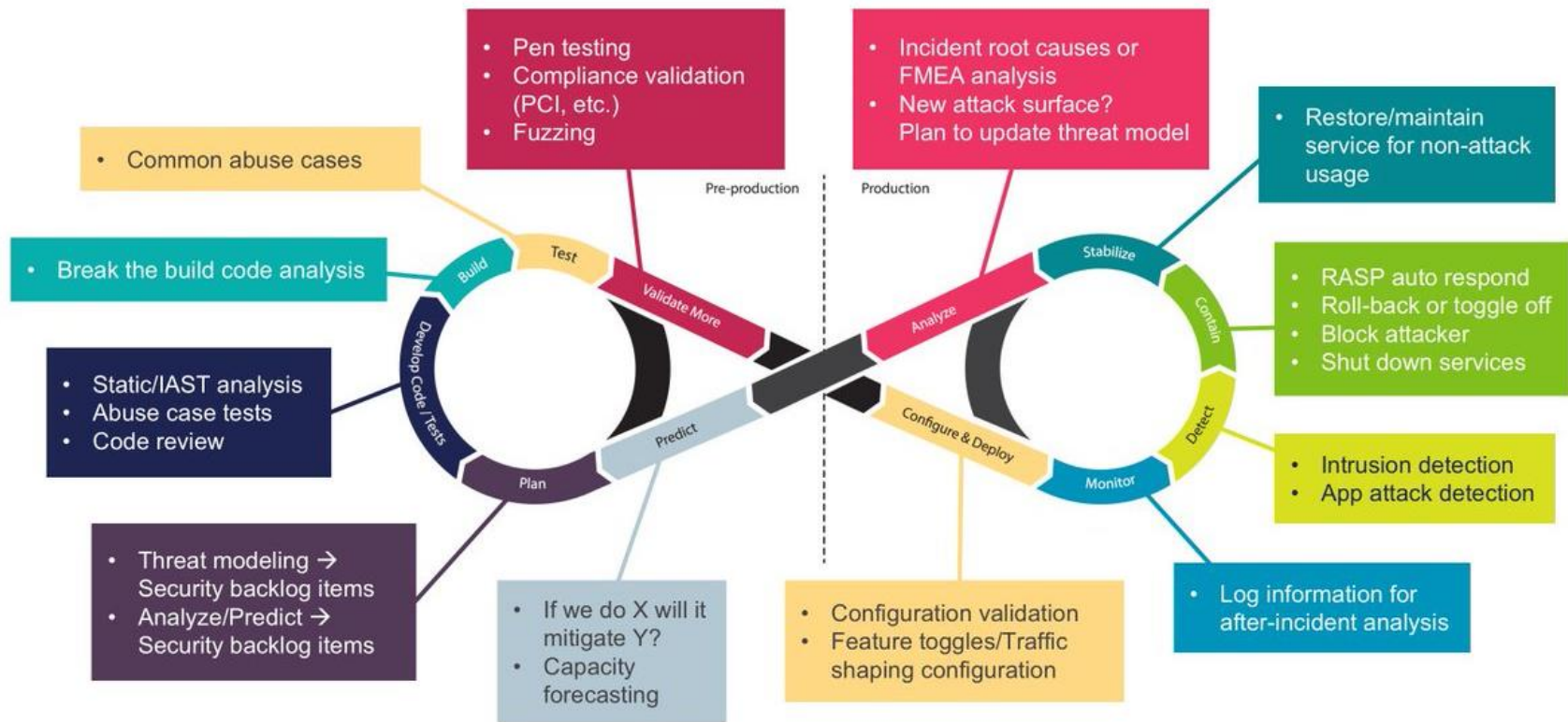
Source: Gartner (October 2017)

面向开发人员整合安全测试工具和流程

对于习惯于强制开发人员遵循已有流程的安全专家来说，首先要转变思维，将持续的安全保证无缝整合到开发人员的持续集成和持续交付工具链和流程中。

- 永远不要让开发人员放弃合作他们自己的工具链环境，包括他们的IDE、BUG追踪和CI/CD工具。
- 设计一种不需要安全人员参与的应用安全扫描模式，要支持自动化，并且对开发人员保持透明。需要安全专家参与只能是极少数情况的
- 要实现能够通过API方式进行自动化和可编程方式调用的安全和合规扫描架构，而不是继续使用安全厂商的工具控制台。测试以及结果应该是自然的方式呈现到开发的仪表板中。
- 对于所有的新应用系统，收集简单的自动化测试需求，同时使用威胁建模工具作为非功能性需求。





放弃在开发阶段解决所有漏洞的“完美”想法

完美是优秀的敌人

在数字化业务年代，追求完美的安全是不现实的，不存在完美的安全，也不存在零风险。安全本身就是一种妥协，应结合风险管理和信任机制来采取适度的方式。

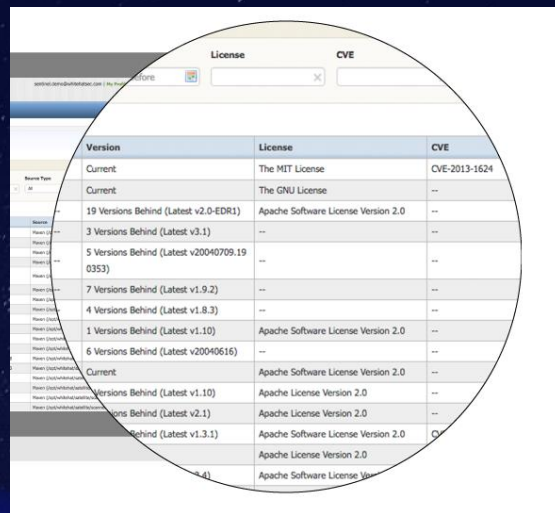
- 需要在效率、误报和漏报三者之间找到一种平衡。
- 开发阶段发现的低风险的问题，可以放到生产阶段，采用其它手段进行补偿性的控制，比如基于网络和主机的入侵防御系统、Web应用防火墙、运行时应用系统自保护（RASP）。
- 安全并非止于开发阶段，生产阶段同样会有问题。开发和生产阶段的安全问题，都应被采取相同的管理策略。
- 最后，要知道安全人员的责任是发现和揭示风险，而如何处置则更多取决于产品或服务所有者。

IMPACT	Risk Management Actions		
Significant <ul style="list-style-type: none"> Financial Loss > \$5MM Stakeholder faith impacted and lasts > 18 months Isolated or Multiple Loss of Life Multiple events of fine, fraud or legal action Complete system crash with loss of critical data Inability to recruit, retain staff to operate Labour disruption that impacts graduation 	Considerable Management Required	Must manage and monitor risks	Extensive management essential
Moderate <ul style="list-style-type: none"> Financial Loss < \$5 MM Stakeholder faith impacted and lasts 6-12 months Significant injury to one or more Isolate incidents of fine, fraud, or legal action System crash during a peak period Difficulties in recruiting and retaining staff Labour disruption that impacts operations of any duration 	Risks may be worth accepting with monitoring	Management effort worthwhile	Management effort required
Minor <ul style="list-style-type: none"> Financial Loss < \$500,000 Stakeholder faith impacted and lasts < 6 months Isolated injury Civil or criminal action threatened System off-line periodically during non-peak periods 	Accept risks	Accept but monitor risks	Manage and monitor risks
	Low > 36 months	Medium 18 to 36 months	High 12 to 18 months
	LIKELIHOOD		

首先聚焦于识别和清除已知的关键漏洞

现代的软件是组装出来的。软件大量使用从公开渠道获得的预先构建好的组件、类库、容器和框架，真正的定制代码只占不大的比例。所以要解决主要风险，只要首先保证在这些类库、框架和组件中的已知安全漏洞和配置不当在投产前已经被解决。

- 首先保证在这些类库、框架和组件中的已知安全漏洞和配置不当在投产前已经被解决。从安全角度看，从已知的代码中识别出已知漏洞要比从定制代码中识别未知漏洞更加容易
- 应建立软件构成分析（SCA，software composition analysis）清单，来识别每一个应用系统中用到的组件，包括但不限于操作系统、数据库、中间件、类库、框架等。SCA可以正向地控制已知风险，也可以在新漏洞发现时提供反向的快速定位。
- 不但要识别代码中的漏洞，也要对其它的安全问题一起进行识别，如恶意软件、敏感数据。
- 要确保不要把密钥、敏感配置信息等放到代码中。

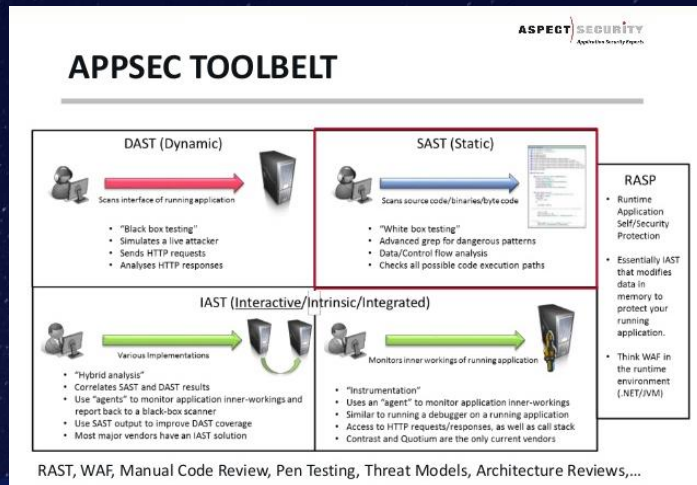


Version	License	CVE
Current	The MIT License	CVE-2013-1624
Current	The GNU License	--
19 Versions Behind (Latest v2.0-EDR1)	Apache Software License Version 2.0	--
3 Versions Behind (Latest v3.1)	--	--
5 Versions Behind (Latest v20040709.19 0353)	--	--
7 Versions Behind (Latest v1.9.2)	--	--
4 Versions Behind (Latest v1.8.3)	--	--
1 Versions Behind (Latest v1.10)	Apache Software License Version 2.0	--
6 Versions Behind (Latest v20040616)	--	--
Current	Apache Software License Version 2.0	--
Versions Behind (Latest v1.10)	Apache License Version 2.0	--
Versions Behind (Latest v2.1)	Apache License Version 2.0	--
Behind (Latest v1.3.1)	Apache Software License Version 2.0	C
	Apache License Version 2.0	--
	Apache Software License	--

除了传统的SAST/DAST，要有新方法和新手段

在DevOps模式下，传统的SAST/DAST并不能很好地满足实际的需要，主要表面在扫描效率和误报两个方面，所以要考虑替换或重构现有的工具，并要考虑引入新的工具如IAST。

- 对现有的SAST/DAST进行一些改进，以更好地适应DevOps的场景
 - 将简单的安全测试能力整合到IDE中；
 - 扫描要自动化，能够通过API方式被开发人员的工具所触发
 - 扫描应该面向开发人员，而不能由安全人员启动、配置和解释。
- 安全人员要基于风险原则给出处理顺序，以简单和直观的形式提供给开发人员，方便他们进行修复。
- 可以考虑交互式应用系统安全测试（IAST，Interactive Application Security Testing）工具。IAST兼具SAST/DAST的能力，但克服了它们的弱点，能显著提高效率和降低误报。



重视对开发人员的培训

开发人员是DevOps的主要角色，要保证应用的安全，必须对他们进行基本的安全知识培训，让他们掌握基本的安全技能，从而能够在工作中降低常规问题出现的概率。

- 开发团队的每个成员都应接受基本的安全培训，只是内容因角色不同而不同。
- 对开发人员进行必要的安全编码技能的培训，比如如何避免OWASP TOP 10
- 培训要全方位、多角度的，既有新人初期培训，也有周期性的培训，还要密切观察开发人员出现的问题并及时给予有针对性的培训。



开发人员就是开发人员，不是安全专家，所以一切更简单和直观

开发人员的内心不会把安全当作首要目标，但也不会完全忽视安全，他们期望事情更简单一些，更直观一些，更准确一些，而不是把时间浪费在似是而非的事务上。所以，安全团队要使安全更简单、直观和准确。


- 清楚地告诉开发人员在什么时候应该干什么，并尽可能地自动化
- 对于大部分的系统，直接提供一个好用的威胁建模工具，让开发人员自行完成威胁收集并得到结果
- 提高扫描工具的精准度，尽可能地减少误报，并提供更直观的报告



采用安全先锋模式

与开发人员相比，安全人员总是比较少的。要实现在资源有效的情况下努力提高开发中的安全水平，可以参考一个方法，就是安全先锋模式（Security Champion Model）。安全先锋相当于安全团队在开发现场的一个代表，可以作为现场的顾问，并协助进行一些策略的执行。

- 每个开发团队都应该有至少一名安全先锋，大的开发团队可能需要更多的安全先锋。
- 确定安全先锋人员的首选方法是志愿，安全团队应该通过一些方法在开发团队中找到对安全有兴趣有热情的人，鼓励他们成为安全先锋；如果找不到合适的人员，只能通过指定的方法产生。
- 将安全先锋视为安全团队的外援，对他们进行专项的培训，让他们了解更多、更全的安全知识和要求。
- 定期举行交流会，加强先锋们之间的交流和学习，安全团队收集他们的反馈。



Security Champion

- Member of the project/product team
- Key contact for security
- Not a security expert
- Mandatory for each project



Security Broker

- Member of the security team
- Ensures security keeps a seat at the table
- Key contact for security champions
- Platform and security by design thinking

不但要关注代码的安全，也要关注环境的安全

在讨论DevOps时，不能忽略基础架构和运行平台相关的内容，“基础架构即代码”要求对基础架构相关的内容（如自动化脚本）采取相应的控制措施，包括版本控制、审计等。

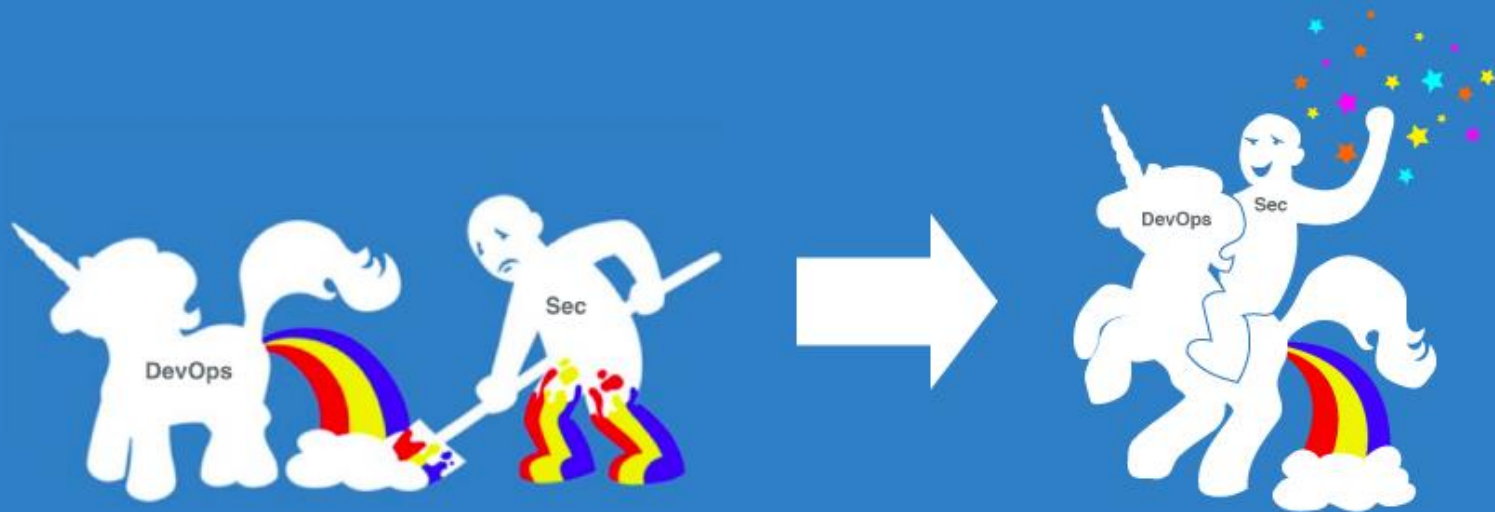
- 系统源代码和运维相关的自动化配置，共同构成了DevOps模式下系统的整体，所以要对运维相关的代码实行与源代码一致的管理策略，包括审计、保护、签名、变更管理、版本管理。
- 容器(Container)是另一种需要采取相似策略的对象。容器和容器管理系统应具有保存变更操作系统的能力。

树立基础架构不可手工更改的意识

应树立一种意识，即基础架构是不可手工更改的。如果要进行一些变更（如安装补丁或配置变更），要回到开发阶段进行，并通过自动化工具进行重新部署。

- 类库、组件，乃至操作系统的镜像，都应从一个安全的仓库中生成出来。运行中的系统，就是安全仓库中的一个副本。如果要更新一个文件，应在安全仓库中进行更新，而不是在生产系统中进行。这样，这个打了补丁的新系统可以通过现有的DevOps的流程进行重新部署和重复部署。这对安全和运维很重要，因为很多运维事件的根源在于某各类型的配置不当、管理不善或补丁缺失。
- 这种模式，除了保证一致性外，还能为快速恢复和重新部署提供可能性。比如，有一个应用的实例被入侵了，那消除影响的最快的方法就是重新部署。

**改变思维，成为服务者和合作者
与开发紧密集成，实现持续交付
抓住主要矛盾和矛盾的主要方面
让事情变得更简单
从头就要安全
努力自动化
赋能**



致谢

FIT

·

REEBUF

THANKS
谢谢

