安全和研发不得不说的那些事

--首届线上技术研讨会

张坤-破天
智联招聘-安全总监
owasp北京分会负责人

Who am i?

Who is the OWASP Foundation?

The Open Web Application Security Project (OWASP) is a nonprofit foundation that works to improve the security of software. Through community-led open source software projects, hundreds of local chapters worldwide, tens of thousands of members, and leading educational and training conferences, the OWASP Foundation is the source for developers and technologists to secure the web.

Who am i?

Open Web Application Security Project

OWASP中国

拥有七千余名会员，共同推动了安全标准、安全测试工具、安全指导手册等应用安全技术在中国的发展。

每个人都可以免费加入OWASP中国，利用免费和开放许可证获得OWASP的相关资源。

关于“owasp”北京分会

000、owasp中国王颉副主席直接领导

001、2019年全国第一届owasp企业安全峰会

010、2019年全国最后一届owasp企业安全峰会

011、owasp北京分会核心会员数百人，两个高活跃微信群

100、建立了良好的社区生态，每日安全资讯、技术讨论、知识共享、攻坚、项目研究等。

本次议程

- 简单的安全研发分析--语大义之方
- 希望每个开发者需要知道的事儿--论万物之理
- 安全设计简要概述--方可受益
- Q+A

OWASP
Open Web Application Security Project

本次议程

- 简单的安全研发分析-语大义之方

  目标和价值

- 希望每个开发者需要知道的事儿-论万物之理
  - 简介描述
  - 评估
  - Build

- 安全设计简要概述--方可受益

- 结论 Q+A

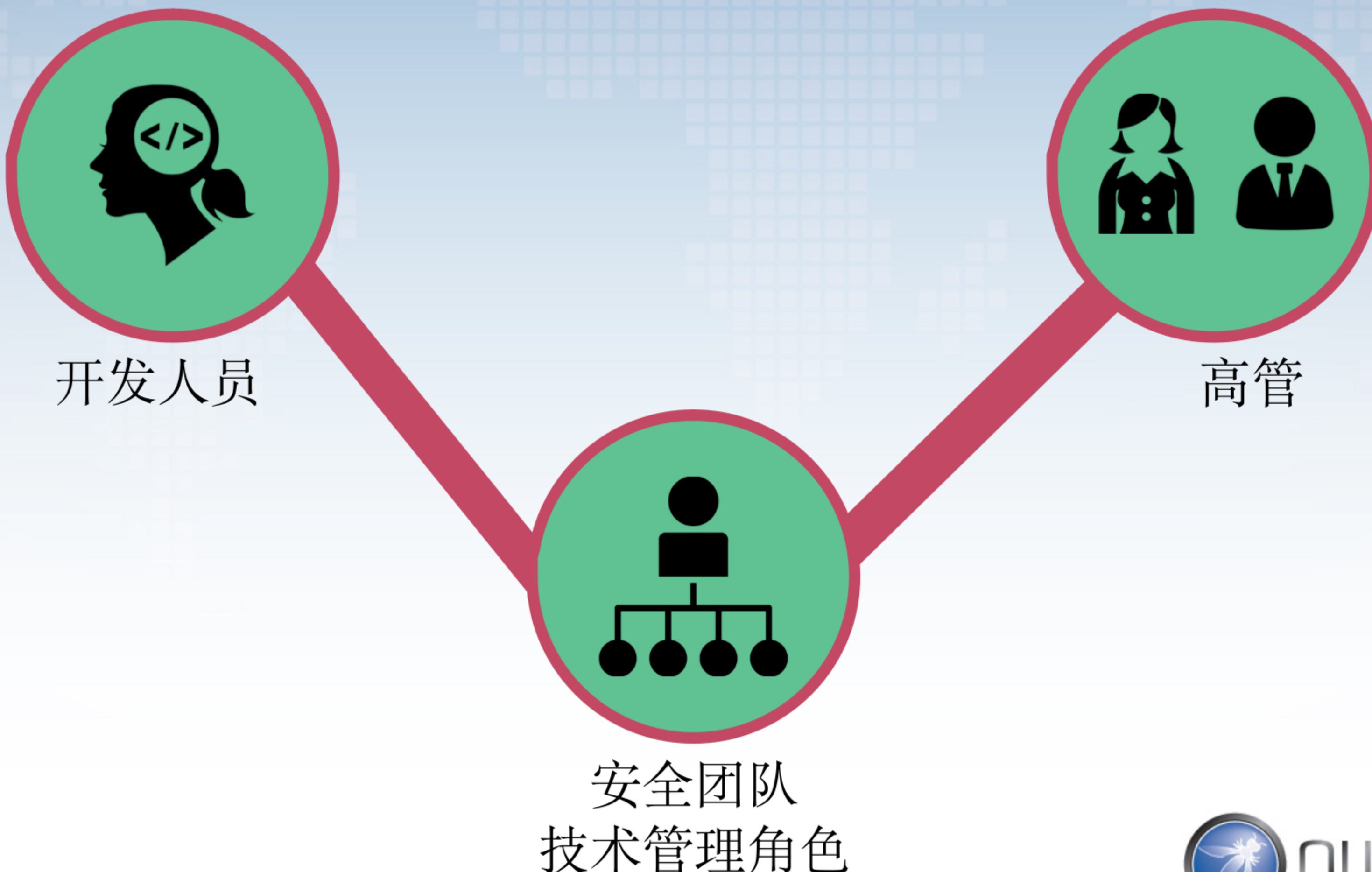
OWASP
Open Web Application
Security Project

在未来

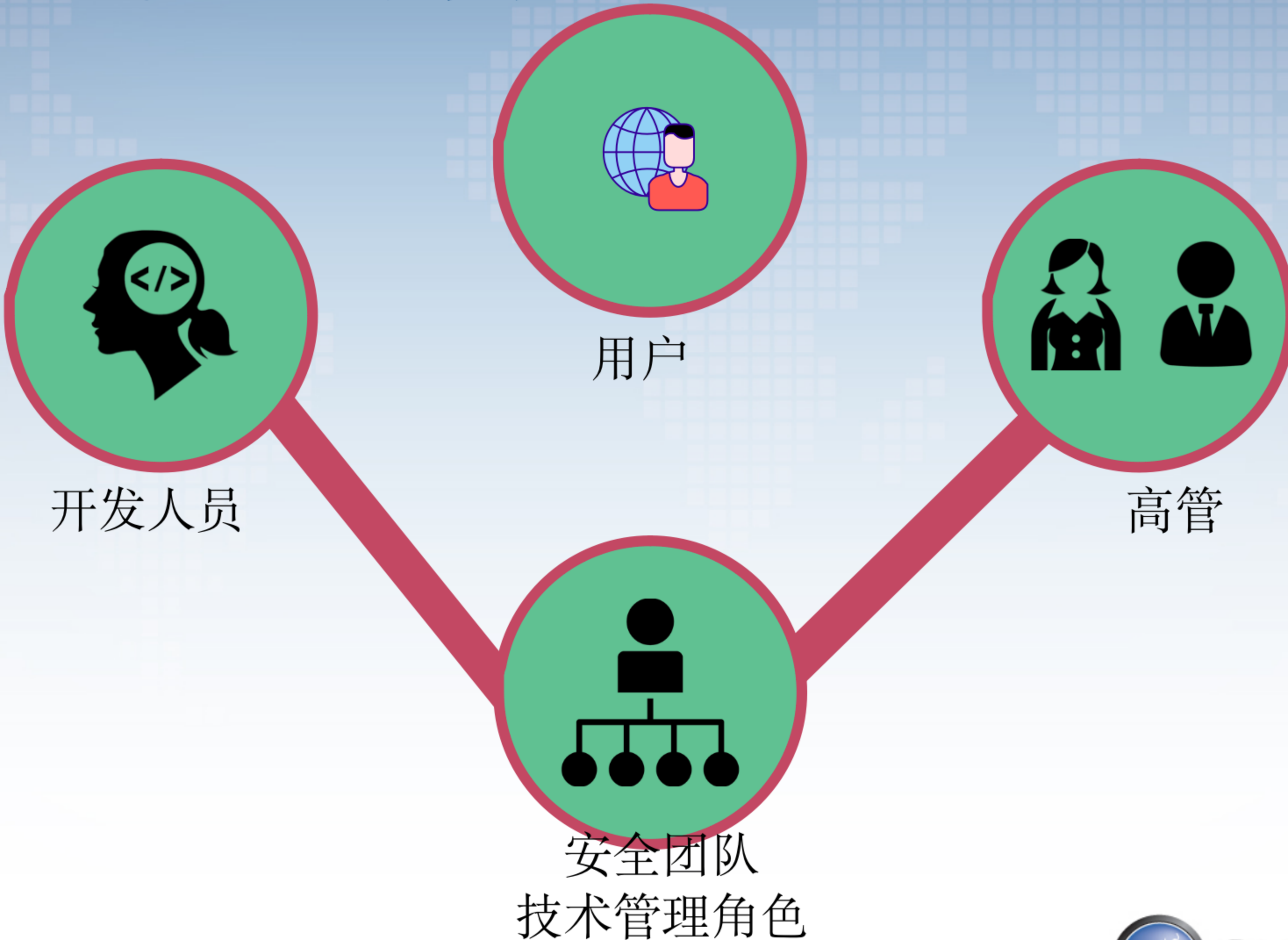
所有“研发者”都将受到安全启蒙

OWASP
Open Web Application Security Project

“启蒙”范围-传统



“启蒙”范围-需求



OWASP
Open Web Application
Security Project

研发安全的目标

1. 降低部署代码中的漏洞数量；
2. 开发安全的软件；
3. 指导开发团队开发安全的软件；
4. 为企业内部的软件安全开发提供标准化的流程和工具；
5. 通过评估和度量，证明企业的软件安全能力成熟度。

安全的价值特征

降低威胁的可能性

保护资产价值

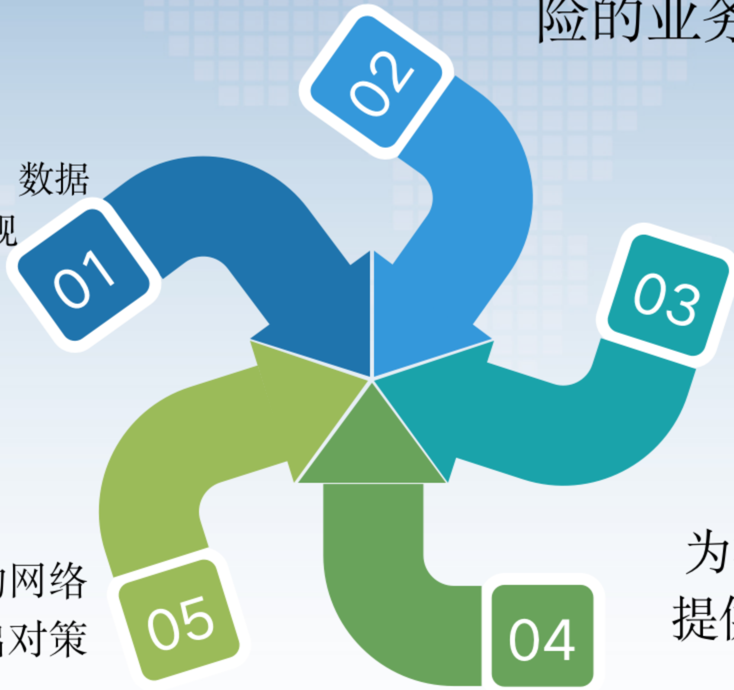
应用安全价值体现

减少漏洞的暴露程度

研发安全的优先级

优先修复有风险的业务漏洞

确保应用符合隐私安全、数据保护和信息安全相关法规



衡量和管理应用安全风险和流程

为构建应用安全提供指导意见

分析针对应用的网络威胁并找出对策

OWASP
Open Web Application Security Project

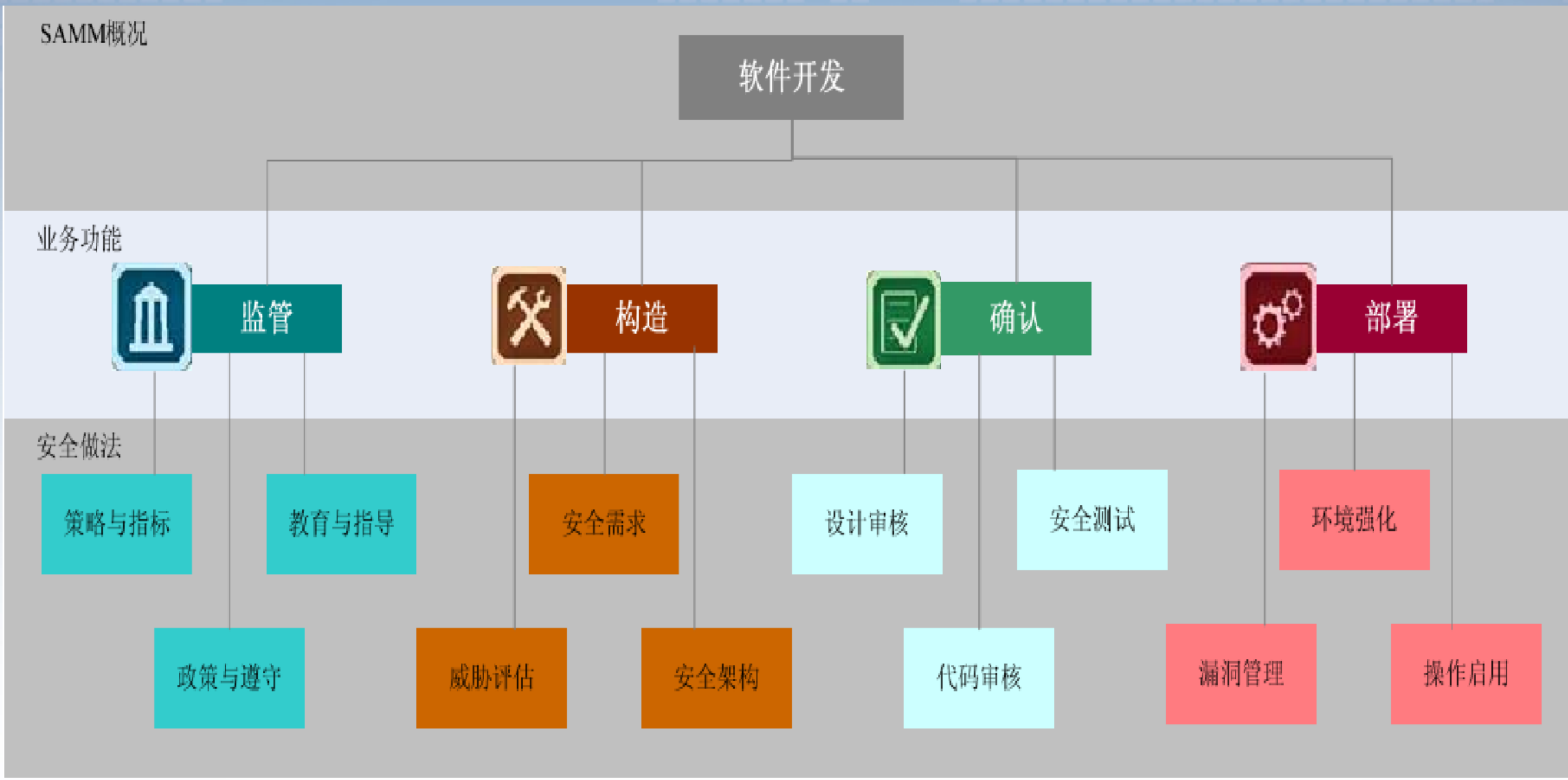
关于产品安全的沟通

将产品安全和每一个产品团队融合

安全团队/Security Champions

本次议程

- 简单的安全研发分析--语大义之方
- 每个开发者都需要知道的事儿、--论万物之理
  - Assess
  - Build
- 安全设计简要概述--方可受益
- 结论 Q+A

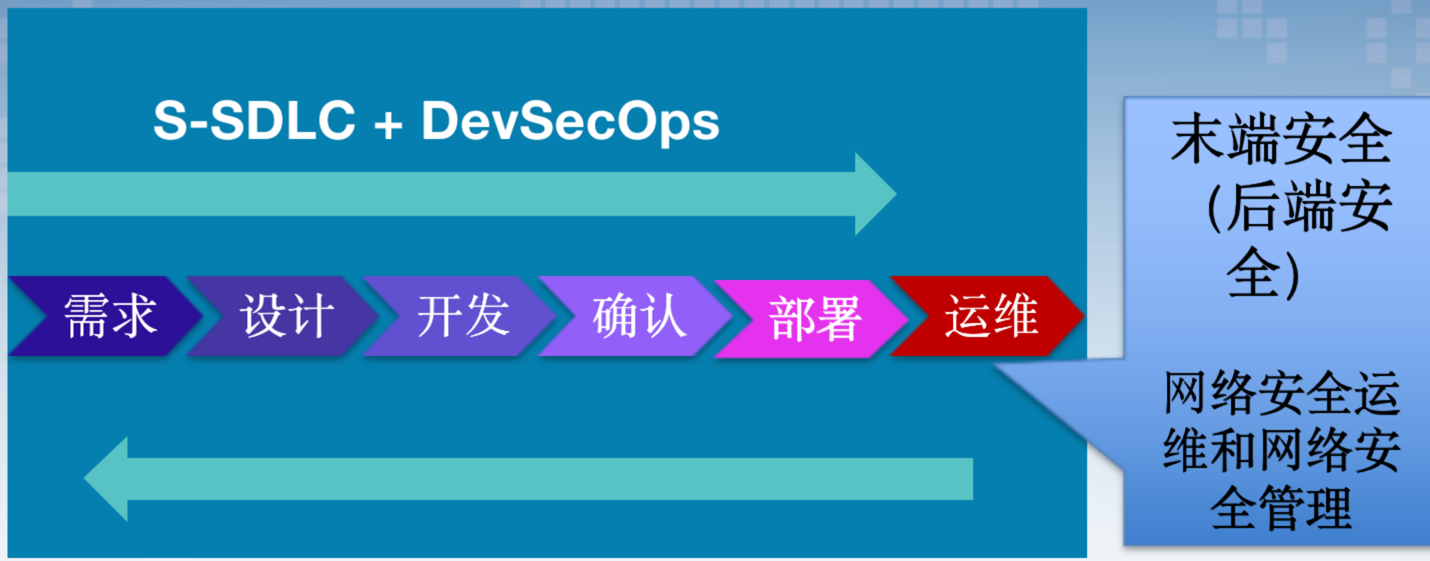


OWASP
Open Web Application Security Project

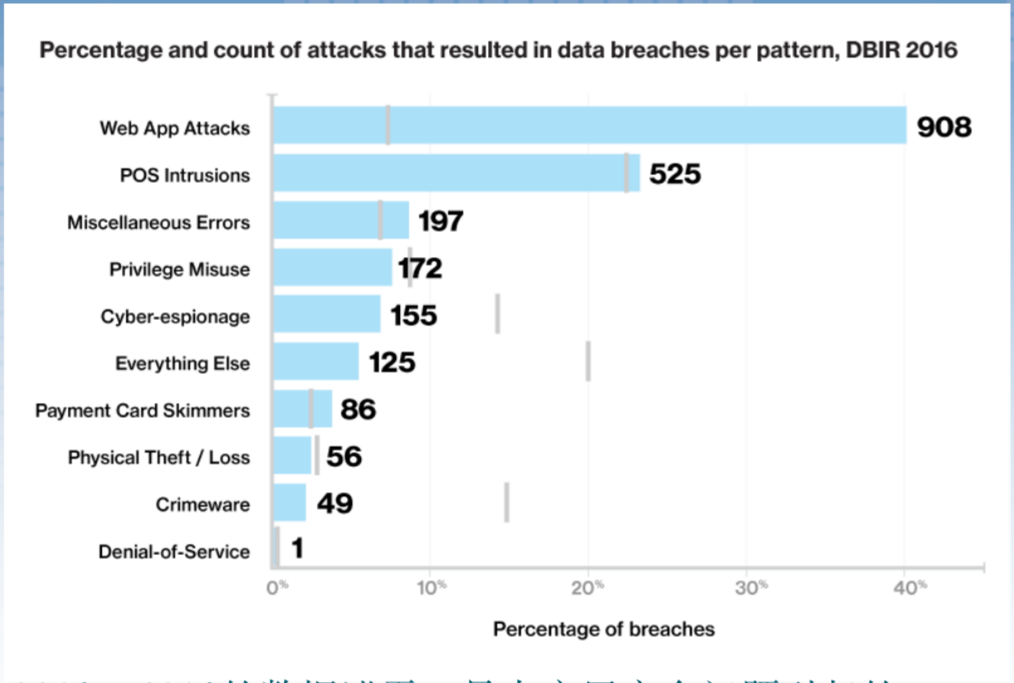
安全“研发”三个关键问题

1. 信息资产和他们的价值及功能
2. 谁是利益相关者和威胁代理
3. 保证资产实现价值和功能需要什么样的安全措施

应用安全的本质原因



把安全左移，从软件系统诞生的源头出发!



29%—40%的数据泄露，是由应用安全问题引起的!

——2017年Verizon 数据泄露调查报告

低劣的应用安全是个问题

做好应用安全的好处

在产品开发过程中，越早修复安全漏洞，成本投入越低。

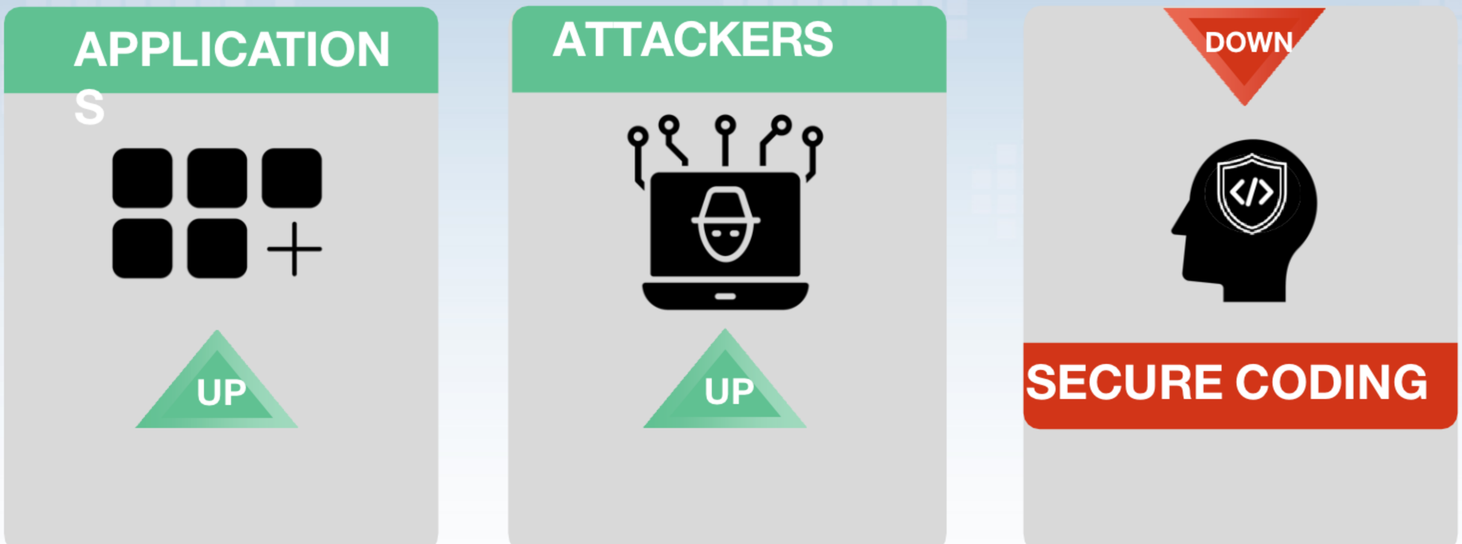
源: Ponemon Institute Research

应用程序安全规划的目标

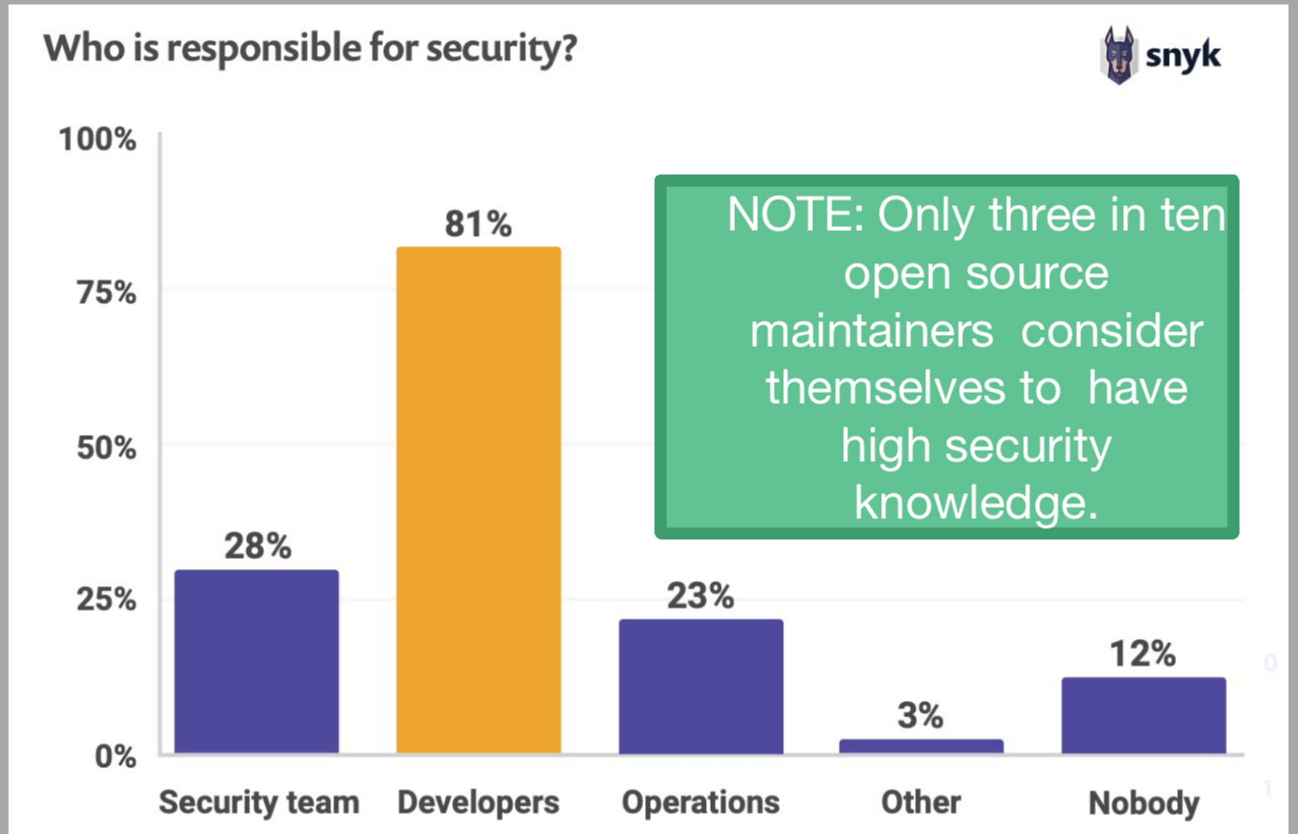
IT IS *NOT*	IT *IS*
实现零风险	快速、及时、有效的发现风险、修复风险
发现、追究开发人员的安全问题	用可衡量的方式提升安全性

OWASP
Open Web Application Security Project

开发人员和产品的安全状态



安全责任: 十分之三的开源维护者认为自己具有高度的安全知识



Source: The state of open source security report 2019, snyk

关注，但是没时间

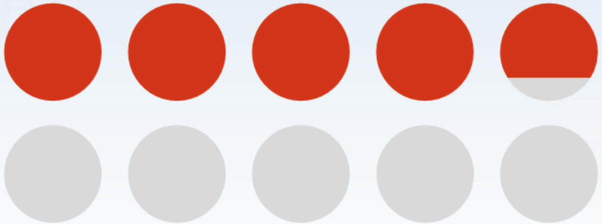
关注开源安全

83%的开发人员担心他们使用的开源代码是否安全。

Source: Enterprise JavaScript in 2019, npm

安全很重要，但时间不足

48%的开发人员表示，他们认为安全性很重要，但没有足够的时间花在安全性上。



Source: DevSecOps Community Survey 2019, Sonatype

OWASP
Open Web Application Security Project

三方代码平台：
开发人员当前的知识来源

"在某些情况下，信誉得分高的用户的不安全建议被选择为接受的答案，而不是信誉得分较低的用户的正确解决方案。"

Secure Coding Practices in Java: Challenges and Vulnerabilities

"在Google Play上可用的1,305,820个Android应用程序中。显示196,403（15%）...包含易受攻击的代码片段，这些片段很可能是从Stack Overflow复制的。"

Stack Overflow Considered Harmful? The Impact of Copy&Paste on Android Application Security

结论：一

CONCLUSION 1

安全知识LOW

CONCLUSION 2

安全关注的是高风险

CONCLUSION 3

安全资源匮乏

CONCLUSION 4

三方平台是安全漏洞的来源

The easy way out

C1 定义安全需求	C2 安全框架和库	C3 安全的数据库管理	C4 编码和转义
C5 验证所有输入	C6 数字身份管理	C7 访问控制	C8 数据保护
C9实施安全日志记录和监视	C10 处理所有错误和异常		

Think at a higher level

常规的安全需求包括:

- 身份认证的安全需求
- 会话安全需求
- 权限控制安全需求
- 日志审计安全需求
- 数据加密安全需求
- 网络以及其他隔离安全需求
- 基础设施安全需求
- 安全编码相关需求

具体的防护机制需要采集的信息有:

- 认证系统相关信息
- 权限控制系统相关信息
- 加密方案以及实现机制
  - 包括密码和凭据的生成，保存方案
  - 业务敏感数据的存储加密方案
  - 传输通道的加密方案
- 日志审计和监控方案
- 网络隔离方案

Ten things I wish every developer knew about security

10 下一代应用安全

9 Docker 和 Kubernetes 并不是安全产品

8 GitHub 不是最好的代码、秘密存储地方

7 DevOps中安全是静默的

6 Shift {left, right, outwards} – 刚开始

OWASP
Open Web Application Security Project

Ten things I wish every developer knew about security

5三方组件和开源漏洞甚多

4 OWASP 可作为安全自研企业的宝库

3 安全团队不能完全模拟黑客攻击方法基于攻击

2 安全和研发应是战友，不是对手

1 每个人都是安全人员，安全需求无处不在

Open Web Application Security Project

#1: 每个人都是安全人员，安全需求无处不在

10 THING

OWASP
Open Web Application Security Project

#1: 每个人都是安全人员，安全需求无处不在

- ASSESS

- 了解组织的安全文化的痛点和核心
- 收集各安全角色意见、建议
- 进行风险评估并发现脆弱点
- 检查开发生命周期的运行

- BUILD

- 获得资源的投入
- 公开传达安全的投入建设现状
- 简化方法，语言和框架选择，并提供适当的安全指导。

OWASP
Open Web Application
Security Project

#2: 安全和研发应是战友，不是对手

当开发人员成功时，
安全团队就会成功，达成有价值的协作关系，向减少脆弱性的方向发展。

10 THING

OWASP
Open Web Application Security Project

#3: 安全和研发应是战友，不是对手

ASSESS

- 开发人员会隐藏或估计避开安全性吗?
- 是否存在安全拥护者计划?
- 有安全培训计划吗?

BUILD

- 开发人员针对安全的同理心
- 安全专员角色的建立
- 安全团队提供安全开发角色配合开发人员
- 安全价值、成就的认可

OWASP
Open Web Application Security Project

#3. 安全团队不能完全模拟黑客攻击

采用的方法基于攻击手法

无论我们投入了多少的精力、成本在测试、攻击、破坏、演练上面，如果没有整合结果进行提升，那么我们在面对真正攻击的时候并不会更好。

10 THING

OWASP
Open Web Application Security Project

#3. 安全团队不能完全模拟黑客攻击

采用的方法基于攻击手法

ASSESS

- 确定组织选择的合适的安全模式.
  - 红蓝对抗,渗透测试、破坏模式?
  - 安全代码的构建者和防御者
- 确定是否存在基于风险的方法

BUILD

- 使用全面的方法来针对开发人员进行基于风险的培训教育:
  - **Builder**: 安全编码
  - **Defender**: red team / blue team / purple team
  - **Breaker**: 靶场juiceshop

OWASP
Open Web Application Security Project

#4: OWASP 可作为安全自研企业的宝库

Top Tens

OWASP Top 10 - 2017
The Ten Most Critical Web Application Security Risks

OWASP ProActive CONTROLS

OWASP API Security Top 10 2019
The Ten Most Critical API Security Risks

Cheat Sheets

OWASP CHEAT SHEET SERIES PROJECT

Tools

DEPENDENCY-CHECK

DEPENDENCY-TRACK

10 THING

OWASP
Open Web Application Security Project

#4: OWASP 可作为安全自研企业的宝库

ASSESS

- 确定多少人知道或了解owasp的存在
-

  验证是否在应用程序安全计划中包含owasp内容

BUILD

- OWASP Top 10 学习培训
- 使开发人员接触安全备忘录
- 集成ZAP和依赖检查

#5: 三方组件和开源漏洞甚多

下载使用带有风险的java组件:

6.1% (2015),

5.5%(2016),

12.1% (2017),

10.3% (2018).

Source: 2019 State Software Supply Chain by Sonatype

11% 被下载的npm模块

2018年具有致命风险

(51% 有风险).

Source: NPM Future of JavaScript

#5: 三方组件和开源漏洞甚多

ASSESS

- 分析第三方组件、开源组件的处理过程和工作方式
- 确认开发人员在编写代码和构建时使用的工具集

BUILD

- 从威胁场景考虑，确保正确的处理
.
  - 已知组件的风险
  - 后门
- 部署SCA工具集并在任何发 现的基础上中断构建。

OWASP
Open Web Application Security Project

#6: Shift {left, right, outwards} – 刚开始

开始向左，开始向右
=安全 开发生命周期

(旧瓶装新酒)

#6: Shift {left, right, outwards} – 刚开始

ASSESS

- 全面分析安全开发生命 周期
- 与Dev + Ops结合以确定SDL的使用。

BUILD

- 向研发、产品等贯彻从左开始的概念。
- 从头开始-研究并实施可提供安全可见性和生产防护的工具。

OWASP
Open Web Application Security Project

#7: DevOps中安全是静默的

没有集成安全性，就无法拥有真正的DevOps。

DEV
SEC
OPS

#7: DevOps中安全是静默的

ASSESS

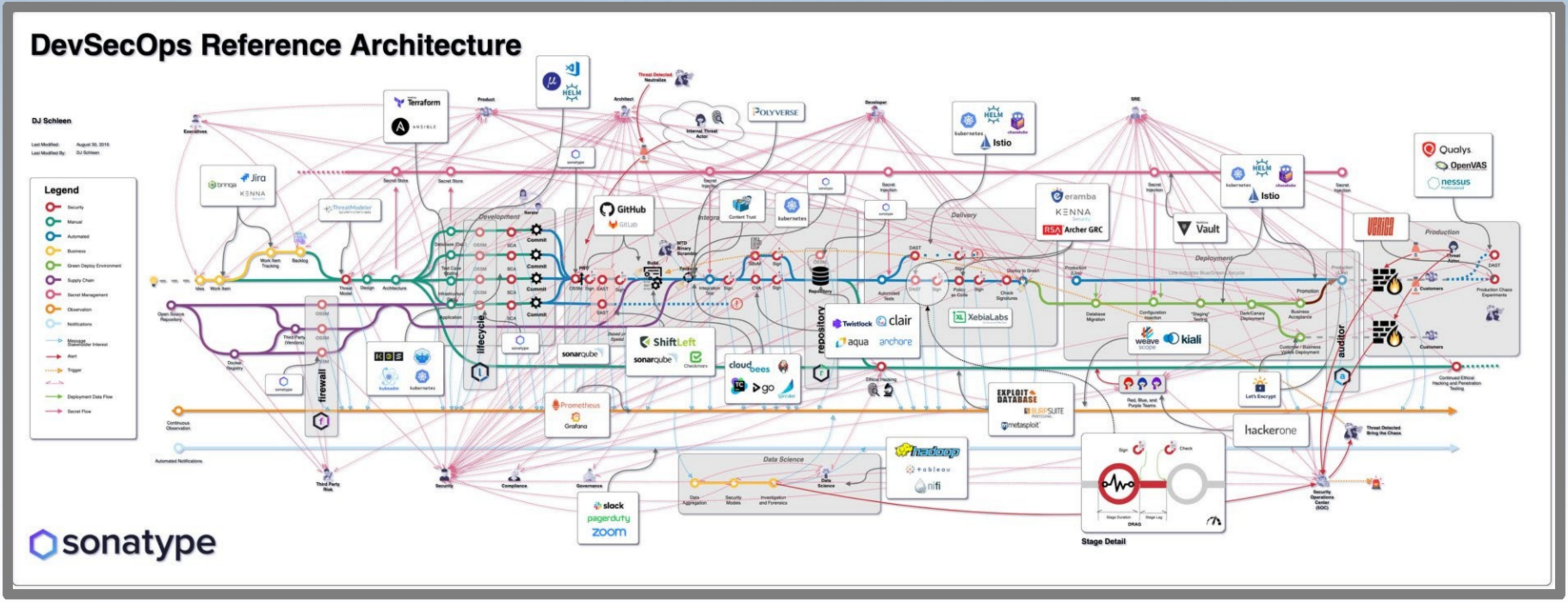
- 开发方法的评估、识别
  - WaterFail, AgileFall, Agile, DevOps
- 考虑不同方法、不同阶段，并在每个步骤中分析安全性的集成。

BUILD

- Cross pollinate and develop DevSecOps Unicorns -- specialists that cross all three barriers.
- 使用其他工具和自动化增强安全性

OWASP
Open Web Application Security Project

#7: DevOps中安全是静默的



Source: https://www.sonatype.com/devsecops-reference-architecture-2019

#8: GitHub 不是最好的代码、秘密存储地方

Secrets

永远不应该出现在我们的源代码管理系统里

10 THING

OWASP

Open Web Application Security Project

#8: GitHub 不是最好的代码、秘密存储地方

ASSESS

- Code review
  从源文件和配置文件中查询secret/ API密钥
-
  使用工具去发现secret和API KEY,比如：gitrob, truffleHog, git- secrets, and OWASP Sedated

SEDATED
Sensitive Enterprise Data Analyzer To Eliminate Disclosure

BUILD

- 在代码管理控制中增加针对secret和API key的安全教育，包含使用和管理
- 使用自动化方式对代码中的secret内容进行识别和发现。

OWASP
Open Web Application Security Project

#9. Docker 和 Kubernetes 并不是安全产品

Will Sargent
@will_sargent

Replying to @edgeroute

They aren't. Even the docker security lead says if you want secure docker you should run it in isolated hypervisor

#9. Docker 和 Kubernetes 并不是安全产品

ASSESS

- 确定体系架构中，对Docker、kubernets的依赖

BUILD

- 了解Docker 和 Kubernetes.的安全相关功能
  - Docker: Namespaces, Capabilities, Seccomp, AppArmor, SELinux
  - Kubernetes: Security Context. Pod Security Policies, RBAC, Namespaces
- 如何安全地使用Docker 和 Kubernetes的指南。

OWASP
Open Web Application Security Project

#10: 下一代应用安全

RASP	IAST	SCA	CWPP
Runtime Application Self Protection	Interactive Application Security Testing	Software Composition Analysis	Cloud Workload Protection Platform

10 THING

OWASP
Open Web Application Security Project

#10: 下一代应用安全

ASSESS

- 确定适合企业的下一代应用程序安全的工具

BUILD

- 在开发和构建中适用IAST
- 在生产运行环境使用RASP
- 使用SCA来检测第三方组件构建
- 使用cwpp保护和审核容器、无服务器的工作环境

OWASP
Open Web Application Security Project

结论：二

CONCLUSION 1

开发人员对安全而言，不是产生问题的根源，是解决问题的办法。

CONCLUSION 2

安全合作关系让技术角色统一并更专注于重要的问题

CONCLUSION 3

“Hacking”不是答案;

CONCLUSION 4

监控依赖关系，调整流程，选择适用的工具

OWASP
Open Web Application Security Project

安全研发的那些事儿

10 下一代应用安全

9 **Docker** 和 **Kubernetes** 并不是安全产品

8 **GitHub** 不是最好的代码、秘密存储地方

7 **DevOps**中安全是静默的

6 **Shift {left, right, outwards} –** 刚开始

5 三方组件和开源漏洞甚多

4 **OWASP** 可作为安全自研企业的宝库

3 安全团队不能完全模拟黑客攻击，采用的方法基于攻击手法

2 安全和研发应是战友，不是对手

1 每个人都是安全人员，安全需求无处不在

OWASP
Open Web Application Security Project

本次议程

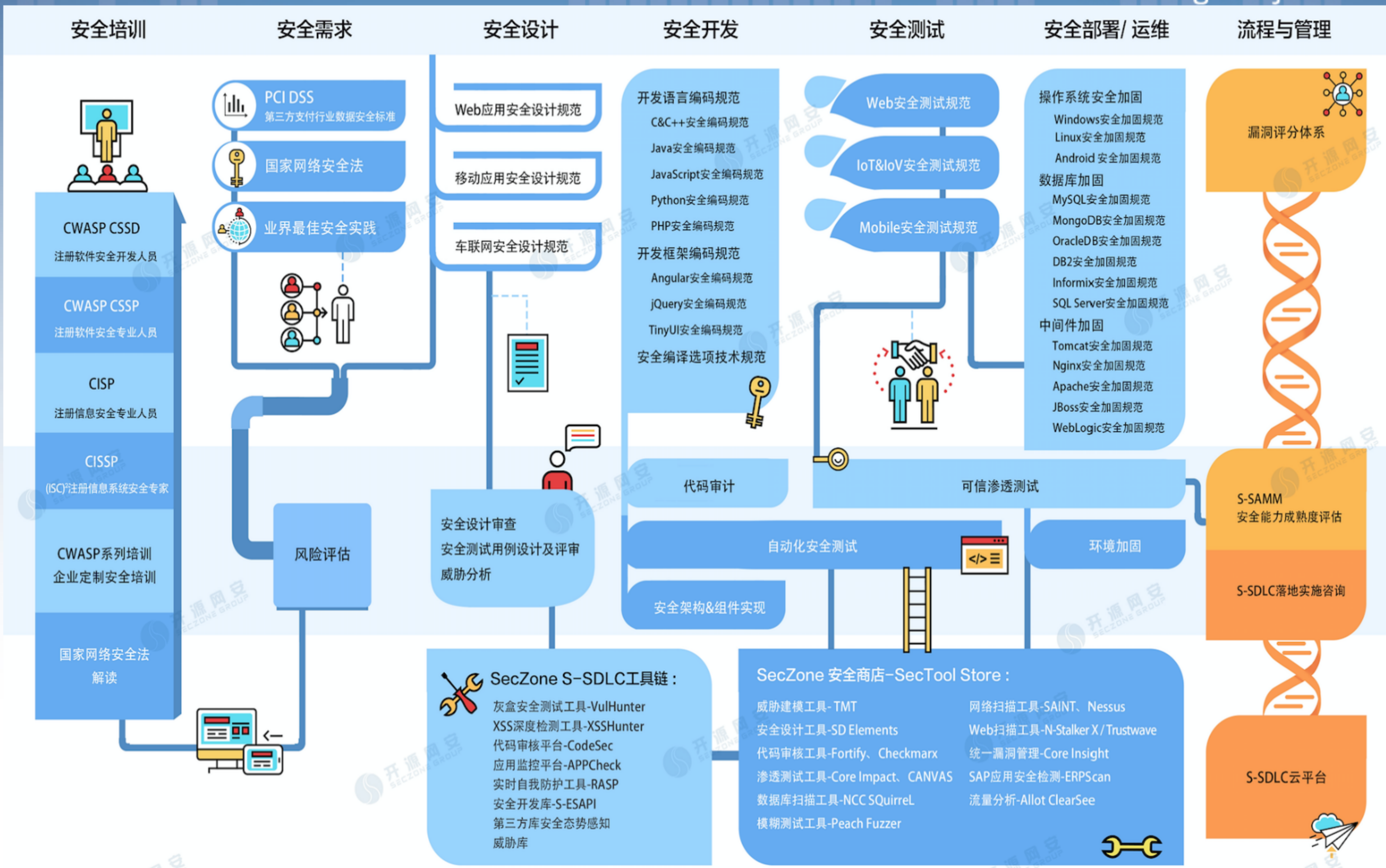
- 简单的安全研发分析-语大义之方
- 希望每个开发者需要知道的事儿-论万物之理
  - 简介描述
  - 评估
  - Build
- 安全设计简要概述--方可受益

talk is cheap, show me the code

- 结论 Q+A

S-SDLC全景图

OWASP
Open Web Application
Security Project



安全设计基本原则

1）最小特权原则

对于请求存储资源的主体，只应该分配最少的必要权限，而且应该保证赋予权限分配的必要时间最短。如果授予一个用户或进程、组件超过其行为必要的权限范围的许可，该用户或进程、组件就有可能获得或修改其没有权限处理的信息。

2）权限分离原则

尽量把软件划分为不同独立的组件，把权限分离成不同的权限许可和认证条件，把用户分离成不同的权限角色。不要将权限一次性授予给一个用户，而是根据需要提供多重的认证与检查机制再进行授予。

3）最少共享机制原则

避免多个主体共享同一个资源，因为敏感的信息可能通过相同的机制在这些主体之间共享导致被其他用户获取。每个主体应该有不同的机制或不同的机制实例，在保证多用户存取访问的灵活性的同时，防止由单一的共享机制导致潜在违背安全性的行为。

Open Web Application Security Project

安全设计基本原则

4）完全中立原则

- 每次主体对资源的请求，系统都应该实行认证和执行检查，特别是和安全相关的内容。
- 以避免错误的赋予主体过高的权限或者在第一次授予权限之后，主体被攻击之后攻击者滥用相关权限。为了提高性能，一些系统会缓存主体的权限，这种做法易使系统具有较高的安全风险。

5）心理可接受度原则

安全机制应该尽可能对用户透明，只引入少量的资源使用障碍，对用户友好，才能在使用时方便用户的理解和使用，真正起到安全防护的作用。如果安全机制妨碍了资源的可用性或使得资源难以获取，那么用户很可能会选择关闭这些安全机制或功能。

6）默认故障处理保护原则

当系统失效或产生故障时，必须是以安全的方式来处理系统信息，系统故障处理默认应该是安全的设置。例如：即使丧失了可用性，也应该保障系统的机密性和完整性；故障发生时必须阻止未授权的用户获得访问权限；发生故障后，应该不向远程未授权的用户暴露敏感信息，如错误号和错误信息，服务器信息之类。

OWASP
Open Web Application Security Project

安全设计基本原则

8）不信任原则

开发者应该假定系统环境是不安全的。减少对用户，外部系统，其他组件的信任，对外部实体所有的输入都需要进行检查，即使对于可信的外部用户输入。另外也不应该信任每次对函数或系统的调用操作都必然会成功，如内存的分配，因此必须对每次函数或系统的调用的返回值进行检查，并进行正确的处理。

安全设计基本原则

7）经济机制原则

复杂性是评估一个系统安全性的重要因素之。如果设计、实现的功能是非常复杂的，那么系统存在安全漏洞的可能性则大大增加，一些问题在复杂的系统中很难被及时发现。系统的设计和实现应该尽量简单，以降低因复杂性带来的安全问题。

OWASP
Open Web Application
Security Project

安全设计基本原则

9）纵深防御原则

软件应该设置多重安全措施，并充分利用操作系统提供的安全防护机制，形成纵深防御体系，以降低攻击者成功攻击的机率和危害。结合多重安全措施使攻击者要绕过每一个机制才能达到目的，提高了攻击者的攻击成本和要求，降低了安全危害。

10）保障最弱环节原则

攻击者一般从系统最薄弱的环节发起攻击，而不是针对已经加固的组件。相对于破解一个数学上已经证明了比较安全的算法，攻击者更喜欢利用软件的安全漏洞。因此软件开发者必须了解自己软件的薄弱点，针对这些弱点实施更强的安全保护措施。

11）公开设计原则

应该假定攻击者有能力获取系统足够的信息来发起攻击，而不是依赖于攻击者不可能知道来保护系统的安全。如果设计的加密算法存在弱密钥，或者系统设有万能口令，等等，攻击者通过反汇编分析能够获取这些信息，攻击者还可能是内部被辞退的员工，因此，依赖于攻击者无法掌握某些特定信息来保护自己的安全是不可靠的。

安全设计基本原则

13）不信任原则

攻击面（Attack Surface），通常也称受攻击面，是指对一个软件系统可以采取的攻击方法集合。因为攻击者对软件的攻击是从其暴露在外部的接口、功能、服务和协议等资源来实施的，通过对每种资源计算其被攻击成功的可能性，并将这些可能性综合归纳，即可以衡量软件的攻击面大小。可以看出，一个软件的攻击面越大，其安全风险也就越大。

减少攻击面是安全设计中的一个重要步骤，软件设计人员需要仔细评估软件中所有的功能模块和接口的特性，分析其可能存在的安全风险，并设定相应的限制措施。如果一个功能/数据接口不是必要的，则应该取消或禁止，或者默认不开启的。如果一个功能/数据接口的配置没有特殊的理由，默认则应该按安全的方式进行设置。

安全设计基本原则

12）隐私保护原则

系统收集到的用户信息都必须实施妥善和安全的保护。攻击者获得了用户的隐私数据之后，可以发起进一步针对用户的各种攻击，如：欺骗等，不应该向其他用户泄露用户的隐私信息。

OWASP
Open Web Application Security Project

五个基础研发应用安全点

1. 所有输入数据都是有害的

■ 直接输入数据：对于用户通过 GET, POST, COOKIE, FILES, ENV, REQUEST输入的数据以及框架提供的数据来源等，即通信协议中从客户端传过来的一切变量，无论是客户手动填写的数据或是客户端浏览器或操作系统自动填写的数据，都可能产生安全问题，需要进行严格的安全性检查。。

■ 间接的输入数据：从数据库、文件、网络、内部API获取数据等，即一些不直接来源于用户，但是又不是程序中定义好的常量数据。比如用户的输入经过层层转化输出到数据库或文件，后面又再次利用的时候，这时变量依然是不可信任的，但往往容易让程序员放松警惕。

2. 不依赖运行环境的安全配置

在编写程序的时候不能有侥幸心理，不能寄希望于配置文件的安全选项；必须将自身的程序置身最不安全的配置下进行考虑。

3. 安全控制措施落实在最后执行阶段

每个安全问题都有其产生的原因。例如SQL注入的原因就是在数据库执行程序提交的SQL语句时产生的问题。因此对于SQL注入安全问题的防范，也需要在数据库操作提交这一瞬间对SQL语句进行过滤和检查才最为恰当。而例如前端问题，就推荐在输出时候进行统一处理，而不是在后端转义。总而言之，对于统一防御的手段，建议系统的底层框架或API中进行总体处理，一个系统的安全处理方式应该统一。

4. 最小化原则

最小化原则适用于所有安全相关的领域，在代码安全方面主要表现为：

■ 用户输入最小化：尽可能少地使用用户的输入。

■ 用户输入范围的最小化：过滤参数的时候尽量使用白名单策略，对于可以明确定义范围的参数要检查参数的有效性，譬如Email，卡号，身份证号等都属于可以明确定义范围的参数。

■ 返回信息最小化：程序错误信息等尽量的对用户屏蔽，不要将原始错误信息直接返回到用户客户端。

5. 失败终止原则

对用户提交的数据进行安全性检查的时候，如果发现数据不符合要求就终止业务的执行，不要试图修正和转换用户提交的参数继续向下执行。在修正和转换的过程中往往会造成不可预知的问题，影响系统安全性。

OWASP
Open Web Application
Security Project

制定一个真正的安全“计划”

- 下周：
  回顾十件事，开始思考所服务的企业接受了多少。
- 三个月：
  制定计划，对开发人员进行有关安全文化主题的教育。
- 六个月内，您应该：
  继续执行计划，并定期与开发人员进行沟通。

OWASP
Open Web Application Security Project

广告时间

- 智联招聘src-https://src.zhaopin.com
- 心意满满：
  - 1、owasp会员享受积分double
  - 2、首发“功守”T恤
  - 3、各种礼品卡、书籍、礼盒、玩偶手办、背包等

OWASP
Open Web Application Security Project

Q&A

https://owasp.org/
www.owasp.org.cn/

scanner

scanner@sina.com

OWASP
Open Web Application
Security Project